PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK SEMI HOT STANDBY ROUTER PROTOCOL PADA PC ROUTER BERBASIS LINUX

TUGAS AKHIR

Disusun oleh :

Nur Aini Rakhmawati
5100.100.028

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2004

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK SEMI HOT STANDBY ROUTER PROTOCOL PADA PC ROUTER BERBASIS LINUX

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Prasyarat
Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui/Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Febriliyan Samopa, S.Kom ,M.Kom.	Royyana Muslim I, S.Kom
NIP. 132 206 858	

SURABAYA
JULI 2004

Katakanlah siapa gigih menempuh keberhasilan
adangkala tergelincir lalu bangkit kembali
adang ia tersesat lalu berhasil mencapai sasaran
adang ia tergores luka lalu pulih kembali

Sekali dua ia gagal tetapi tidak meletakkan senjata
Tidak berputus asa atau kehilangan cahaya harapan

Tidaklah seorang Muslim menderita karena kesedihan, kedukaan, kesusahan, kepayahan, penyakit dan gangguan duri yang menusuk tubuhnya kecuali dengan itu Allah akan mengampuni dosa-dosanya (HR. Bukhari)

ABSTRAK

Router memegang peranan penting dalam sebuah jaringan. Jika router mengalami suatu masalah dapat berakibat terhambatnya lalu lintas data dalam suatu jaringan. Untuk mengatasi masalah sebuah router, Cisco mengembangkan suatu sistem Hot Standby Router Protocol (HSRP). Namun HSRP ini belum dimiliki oleh router komputer. Padahal harga komputer lebih murah daripada router hardware.

Semi HSRP merupakan suatu metode yang diimplementasikan pada router komputer dengan keterbatasan yang dimiliki oleh sebuah komputer dimana sebuahrouter komputer tidak dapat mengirimkan pesan resign yang mengijikan router lain untuk menggantikannyaketika router komputer ini mengalami kegagalan.

Dalam semi HSRP dibentuk suatu kelompok router yang memiliki subnet yang sama pada masing-masing interfacenya.Pada masing-masing router dijalankan aplikasi yang berfungsi untuk mengatur jalannya mekanisme HSRP. Aplikasi ini memungkinkan router-router tersebut dapat berhubungan untuk mengetahui apakah ada masalah yang terjadi pada router utama yang berfungsi untuk meneruskan data klien dengan menggunakan alamat virtual IP dan MAC. Dalam berhubungan router-router tersebut menggunakan komunikasi broadcast. Ketika router utama mengalami kegagalan maka akan terjadi pemilihan router berdasarkan prioritas yang dimiliki oleh masing-masing router. Router dengan prioritas tertinggi akan menjadi router utama baru. Router yang terpilih tersebut melakukan penambahan IP yang digunakan sebagai virtual IP yang dikenali klien dan mengubah alamat MAC yang sama dengan router utama sebelumnya.

HSRP pada router hardware dapat diimplementasikan dengan semi HRSP pada routerkomputer. Sebagaimana halnya HSRP Cisco perpindahan router dapat dilakukan secara otomatis dengan perpindahan alamat IP dan MAC. Perpindahan router ini tidak dipengaruhi besarnya lalu lintas jaringan dan jumlah router yang terdapat dalam satu kelompok.

Kata kunci: Router, HSRP, MAC, IP

KATA PENGANTAR

Segala puji syukur kehadirat Allah Subhanahu Wa Ta'ala, karena hanya dengan kehendak dan kuasa-Nya, penulis dapat menyelesaikan pembuatan tugas akhir yang berjudul "**PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK SEMI HOT STANDBY ROUTER PROTOCOL PADA PC ROUTER BERBASIS LINUX**".

Tugas akhir dengan beban 4 SKS ini disusun dan diajukan sebagai salah satu syarat untuk menyelesaikan program Strata Satu (S1) pada jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis menyadari bahwa dalam pembuatan tugas akhir ini,masih terdapat kekurangan-kekurangan. Untuk itu saran dan kritik membangun sangat penulis harapkan demi kesempurnaan tugas akhir ini. Akhir kata, penulis berharap tugas akhir ini dapat memberikan manfaat bagi semua pihak.

Surabaya, Juli 2004

Penulis

UCAPAN TERIMA KASIH

Pengerjaan dan penyusunan Tugas Akhir ini tak lepas dari bantuan, bimbingan dan pengarahan dari berbagai pihak. Untuk itu perkenankanlah penulis mengucapkan terima kasih kepada :

1. Bapak dan Ibu tersayang. Sejuta kata terima kasih belum mampu membalas jasa beliau berdua.
2. Bapak Febrilian Samopa, S.Kom, M.Kom dan Bapak Royyana Muslim I, S.Kom atas bimbingan selaku Dosen Pembimbing dan pemberi arahan serta pengalaman yang berharga dalam dunia jaringan dan hardware. Penulis memohon maaf jika selama bimbingan merepotkan beliau berdua.
3. Mas Hermono dan Mas Gayuh yang telah membantu dan menemani penulis dalam suka dan duka di CS-NET.
4. Bapak Wahyu Suadi, S.Kom, M.Kom, atas bimbingannya dalam mempelajari kernel, *library* dan pemrograman dalam linux.
5. Bapak Imam Kuswardayan, S.Kom, yang selalu mengingatkan penulis waktu sholat dan memberikan konsumsi.
6. Bapak Rully Soelaiman, S.Kom. M.Kom, yang dengan gratis berbagi semangat pengerjaan TA.
7. Bapak M.Husni, S.Kom, M.Kom, selaku dosen wali penulis yang telah membimbing penulis selama menjalani perkuliahan di jurusan ini.
8. Nindi, yang telah memberikan perhatiannya selama ini. Semoga persahabatan ini abadi hingga di akhirat nanti.

9. Mbak Num, Didi, Shofie, Fetty, Rahmi, Dik Chang, Dik Riza, Dik Dwi, Dyah L. dan Ruli, yang telah membantu dalam penyusunan buku ini.
10. Bonbun, Ika, Win, Tisna, Mbak Hen dan seluruh panitia Muslimah Ngoprek yang telah memberikan arti bagi penulis.
11. Sokam, yang telah menurunkan ilmu Linux kepada penulis.
12. Mas Tong, Mbak Catur, Mbak Peni, Mbak Rin, Mas Eko dan 'pasangannnya' beserta keponakan yang lucu-lucu, yang telah menemani penulis dalam menjalani kehidupan.
13. Mas Kendy, Mas Aby, Mas Riz, JP, Jakka, Joko, Rozi, Gunna, Mas Ade, Mas Indie, Mas Hindrawan, Mas Ajun, dan seluruh admin Labprog serta Kamal dan Udin yang telah membantu dan memberi semangat penulis.
14. Bapak Roesmanto selaku Pimpinan Redaksi Majalah INFOLINUX, pembaca INFOLINUX terutama Dik Ryan dan teman-teman dalam milis tanya-jawab linux serta wanita-wanita yang tergabung dalam LinuxChix terutama Val Henson yang mendorong penulis agar bermanfaat bagi orang lain.
15. Seluruh teman noceng, yang telah menemani penulis menjalani kehidupan kuliah.
16. Bapak dan Ibu Amin beserta penghuni J-23 yang telah menjadi keluarga penulis selama merantau di Surabaya.
17. Serta segenap pihak yang tidak dapat disebutkan satu persatu yang telah membantu penulis dari awal kuliah hingga tersusunnya tugas akhir ini.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Router merupakan salah satu bagian yang penting dalam sebuah jaringan komputer dimana router dapat menghubungkan dua atau lebih jaringan yang memiliki subnet berbeda dengan topologi yang sama. Namun karena beberapa hal router tidak dapat menjalankan fungsinya dengan sebagaimana mestinya, misalnya : kabel terlepas, kabel rusak, dan mati. Kejadian ini dapat menghambat lalu lintas data dalam subnet jaringan yang berbeda dan dari subnet tersebut ke luar LAN untuk beberapa waktu sampai router tersebut dapat segera diperbaiki.

Untuk mengatasi masalah tersebut, Cisco sebagai salah satu perusahaan yang memproduksi router membuat suatu sistem yang disebut *Hot Standby Router Protocol*(HSRP). HSRP merupakan suatu sistem dimana terdapat kelompok router yang bergabung dengan satu pemimpin router dan yang lain sebagai router pengganti atau dapat kita sebut sebagai router *standby*. Jika pemimpin router ini mengalami kerusakan maka salah satu dari router *standby* otomatis menggantikannya.

Beberapa router dalam bentuk perangkat keras bermerek seperti Cisco, Intel, Proteon, Bay Network, 3Com beberapa diantaranya telah menerapkan HSRP. Namun harga router tersebut yang relatif tinggi sehingga tidak terjangkau oleh perusahaan kecil.

Pada saat ini telah tersedia router yang dibuat dari sebuah komputer dengan sistem operasi Linux. Router dengan sistem operasi linux ini dapat dibuat dengan komputer yang memiliki spesifikasi hardware cukup rendah. Namun router tersebut belum memiliki kemampuan HSRP. Oleh karena itu Tugas Akhir ini membuat aplikasi yang menerapkan HSRP pada router komputer dengan sistem operasi linux.

1.2 Permasalahan

a. Bagaimana mendesain suatu sistem yang dapat melakukan *mekanisme Hot Standby Router Protocol* (HSRP) dengan keterbatasan sebuah komputer dengan sistem operasi Linux.
b. Bagaimana mendesain dan mengimplementasikan ke dalam sebuah jaringan komputer.

1.3 Batasan Masalah

Dari permasalahan-permasalahan di atas, maka batasan dalam tugas akhir ini adalah:

a. Pembuatan perangkat lunak ini dibuat pada komputer dengan sistem operasi linux.
b. Uji coba ini HSRP menggunakan dua buah komputer yang masing-masing bertindak sebagai router *standby* dan router aktif.
c. Router ini tidak menangani terputusnya data klien pada router aktif yang gagal dan hal tersebut diserahkan pada mekanisme yang seharusnya terjadi pada suatu jaringan.

d. Tugas Akhir ini tidak membahas perbandingan HSRP dengan sistem lain

1.4 Tujuan dan Manfaat

Tujuan pembuatan Tugas Akhir ini adalah membuat suatu perangkat lunak yang memiliki prinsip seperti HSRP pada Cisco pada komputer dengan sistem operasi Linux dan mengimplementasikannya dalam sebuah jaringan komputer sehingga kegagalan sebuah router pada jaringan dapat teratasi.

1.5 Metodologi

Pembuatan tugas akhir ini dilakukan dengan mengikuti metodologi sebagai berikut:

1.5.1 Studi literatur

Pada tahap ini dipelajari prinsip penerapan HSRP pada Cisco, cara-cara mengembangkan perangkat lunak pada sistem operasi Linux terutama pemrograman *socket* dan kernel serta pengetahuan lain yang berhubungan jaringan komputer. Pembelajaran ini didapat baik dari buku-buku literatur, paper, manual Linux maupun beberapa artikel di internet.

Pada tahap ini juga akan dipelajari metodologi dan algoritma yang akan digunakan dalam pembuatan perangkat lunak sehingga membantu pada tahap perancangan dan pembuatan perangkat lunak.

1.5.2 Perancangan hardware dan instalasi sistem operasi Linux

Untuk membuat sebuah jaringan komputer dibutuhkan beberapa komputer yang saling terhubung dalam satu jaringan. Sehingga perlu dipersiapkan minimal 4 buah komputer dengan kelengkapan hardaware agar dapat terhubung dengan jaringan. Pada tahap ini juga dilakukan instalasi komputer tersebut dengan sistem operasi Linux dimana ada komputer yang dipersiapkan khusus sebagai router dan ada pula yang dipersiapkan sebagai klien.

1.5.3 Perancangan Algoritma

Untuk membuat suatu sistem HSRP dibuat sebuah algoritma. Algoritma ini meniru prinsip kerja HSRP pada Cisco.

1.5.4 Pembuatan Perangkat Lunak

Pada tahap ini, dilakukan implementasi perangkat lunak berdasarkan rancangan yang telah dibuat pada tahap sebelumnya.

1.5.5 Uji Coba dan Evaluasi

Pada tahap ini aplikasi yang telah dibuat diuji dengan *parameter* sebagai berikut :

a. Kemampuan router aktif dalam mengirimkan pesan ke router *standby*
b. Kemampuan router *standby* menggantikan router aktif.

1.5.6 Penyusunan Buku Tugas Akhir

Pada tahap terakhir ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir. Dokumentasi ini juga dibuat sehingga bagi orang lain

yang ingin mengembangkan sistem tersebut bisa mempelajari dari dokumentasi tersebut.

1.6 Sistematika Pembahasan

Sistematika pembahasan mengenai perangkat lunak yang dibuat dalam Tugas Akhir ini disusun dalam beberapa bab, yang dijelaskan sebagai berikut :

BAB I Pendahuluan

Bab ini berisi pendahuluan yang terdiri dari latar belakang, tujuan dan manfaat, permasalahan, metodologi pembuatan tugas akhir dan sistematika laporan.

BAB II Dasar Teori

Bab ini memuat konsep dan teori pembelajaran yang menjadi landasan pembuatan Tugas Akhir ini. Dasar teori ini meliput pengertian tentang TCP/IP, pengalamatan IP, ARP, MAC, UDP, IOCTL, *socket*, CRC,router dan *Hot Standby Router Protocol* berdasarkan RFC2281 yang dikeluarkan oleh Cisco.

BAB III Desain dan Implementasi Sistem

Bab ini menguraikan bagaimana perancangan sistem Tugas Akhir ini yang meliputi deskripsi singkat aplikasi ini, komponen sistem dan mekanisme sistem ini.

BAB IV Uji Coba dan Evaluasi Sistem

Bab ini berisi bagaimana implementasi dari aplikasi yang telah dibuat berdasarkan desain sebelumnya. Di bab ini juga dijelaskan mengenai hasil uji coba serta evaluasi terhadap sistem tersebut.

BAB V Kesimpulan dan Saran

Bab ini terdiri dari dua bagian, yaitu kesimpulan dan saran. Bagian kesimpulan berisikan kesimpulan yang diambil oleh penulis dan bagian saran berisikan saran-saran yang diberikan oleh penulis untuk pengembangan Tugas Akhir ini selanjutnya.

BAB II

DASAR TEORI

2.1 TCP/IP

2.1.1 Definisi TCP/IP

Dalam berkomunikasi, komputer memerlukan suatu protokol yaitu suatu aturan untuk mengatur proses pengiriman data termasuk proses inisialisasi, verifikasi, cara memulai dan memutuskan komunikasi. Salah satu protocol yang sering digunakan adalah TCP/IP.

TCP singkatan dari *Transmission Control Protocol* dan IP singkatan dari *Internet Protocol*. TCP melakukan transmisi data per segmen, artinya paket data dipecah dalam jumlah yang sesuai dengan besaran paket, kemudian dikirim satu persatu hingga selesai. Pengiriman data akan dibungkus dalam paket dengan label berupa alamat IP si pengirim dan alamat IP penerima.

Hampir seluruh hardware dan sistem operasi menerapkan protokol ini diantaranya Novel Netware, Mainframe IBM, Sistem Digital VMS, Server Microsoft Windows NT, Workstation UNIX, Linux dan FreeBSD.

2.1.2 Sejarah TCP/IP

Internet Protocol dikembangkan pertama kali oleh *Defense Advanced Research Projects Agency* (DARPA) pada tahun 1970 sebagai awal dari usaha untuk mengembangkan protokol yang dapat melakukan interkoneksi berbagai jaringan komputer yang terpisah, dimana masing-masing jaringan tersebut

menggunakan teknologi yang berbeda. Protokol utama yang dihasilkan proyek ini adalah *Internet Protocol* (IP). Riset yang sama dikembangkan pula yaitu beberapa protokol *level* tinggi yang didesain dapat bekerja dengan IP. Yang paling penting dari proyek tersebut adalah *Transmission Control Protocol* (TCP) dan semua grup protokol diganti dengan TCP/IP *suite*.

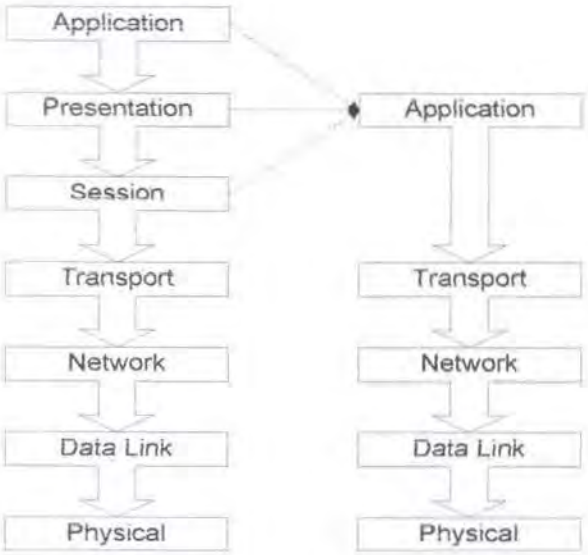
Selain DARPA, TCP/IP juga dikembangkan adalah *Department of defense* (DoD). DoD merasakan kebutuhan akan suatu komunikasi di antara berbagai variasi komputer yg telah ada. Komputer-komputer DoD ini seringkali harus berhubungan antara satu organisasi peneliti dengan organisasi peneliti lainnya, dan harus tetap berhubungan sehingga pertahanan negara tetap berjalan selama terjadi bencana, seperti ledakan nuklir. Oleh karenanya pada tahun 1969 dimulailah penelitian terhadap serangkaian protokol dimana protokol jaringan komputer tersebut yang sama sekali tidak terikat pada jenis komputer maupun media komunikasi yang digunakan.

Pada awalnya TCP/IP sebagai hasil penelitian tersebut dapat berjalan dengan baik karena hanya menyediakan pelayanan dasar seperti *file transfer, electronic mail* dan *remote logon*. Namun ketika beberapa komputer dalam sebuah departemen menggunakan TCP/IP bersamaan dengan protokol lain dalam suatu LAN tunggal dimana komponen IP menyediakan *routing* dari departmen ke *network enterprise*, kemudian ke jaringan regional dan akhirnya ke global internet, menyebabkan jaringan komunikasi dapat rusak. Untuk mengatasinya, DOD mendesain TCP/IP yang dapat memperbaiki dengan otomatis apabila ada *node* dan saluran telepon yang gagal.

Jaringan komputer menggunakan TCP/IP kini semakin berkembang seiring berkembangnya internet. Saat ini TCP/IP merupakan standard pada sistem operasi UNIX dengan disertakan socket library untuk programmer di UNIX mengakes langsung ke TCP socket.

2.1.3 Konsep Lapisan TCP/IP

TCP/IP dibuat merujuk pada model *Open System Interconections* (OSI). OSI merupakan protokol yang dibuat oleh *International Standardization Organization* (ISO) yang merupakan badan dunia yang menangani masalah standarisasi protokol.



Gambar 2. 1 Lapisan OSI dan TCP/IP

Meskipun TCP/IP merujuk pada model OSI, TCP/IP hanya terdiri atas lima lapisan kumpulan protokol yang bertingkat, sedangkan model OSI terdiri atas tujuh lapisan. Kelima lapisan tersebut adalah *application*, *transport*, *network*, *data link* dan *physical*. Lapisan *transport*, *network*, *data link* dan *physical* TCP/IP

sama dengan lapisan OSI, sedangkan lapisan *application* pada TCP/IP merupakan gabungan tiga lapisan OSI, yaitu *application*, *presentation* dan *session* (lihat gambar 2.1).

2.1.3.1 Lapisan Physical

Lapisan ini merupakan media penghubung untuk mengirimkan informasi digital dari satu komputer ke komputer lainnya yang secara fisik dapat kita lihat karena lapisan ini merupakan perangkat keras berupa kabel, Ethernet, *frame relay*, *Token ring*, ISDN, jaringan ATM, radio, satelit atau alat lain yang dapat mentransfer data dari sistem ke sistem.

2.1.3.2 Lapisan Data Link

Fungsi dari lapisan ini untuk mengatur hubungan antar komputer melalui lapisan *physical* yang ada. Karena tugasnya ini, protokol pada lapisan ini mampu menerjemahkan sinyal listrik menjadi data digital yang dapat dimengerti komputer.

2.1.3.3 Lapisan Network

Lapisan ini bertanggung jawab dalam proses pengiriman paket ke alamat yang tepat. Lapisan ini melakukan *encapsulation* paket dari *transport layer* ke dalam IP *datagram* dan menggunakan algoritma *routing* untuk menentukan kemana *datagram* harus dikirim. Masuknya *datagram* diproses dan diperiksa kesahannya sebelum melewatinya pada lapisan *transport*.

Pada lapisan ini terdapat tiga macam protokol, yaitu IP, ARP dan ICMP. IP (*Internet Protocol*) berfungsi untuk menyampaikan paket data ke alamat yang

tepat. ARP (*Address Resolution Protocol*) ialah protokol digunakan untuk menemukan alamat hardware dari *host* atau komputer yang terletak pada network yang sama. Sedangkan ICMP (*Internet Control Message Protocol*) ialah protokol yang digunakan untuk mengirimkan pesan dan melaporkan kegagalan pengiriman data.

2.1.3.4 Lapisan Transport

Lapisan ini bertanggung jawab untuk berkomunikasi antara aplikasi. Selain itu, juga mengatur aluran informasi dan menyediakan pemeriksaan *error*.

Pada lapisan ini data dibagi kedalam beberapa paket yang dikirim ke lapisan *network* dengan sebuah *header*. *Header* mengandung alamat tujuan, alamat sumber dan *checksum*. *Checksum* berfungsi untuk memastikan data tidak mengalami kerusakan atau hilang.

Lapisan ini mengandung dua protocol yaitu TCP (*Transmission Control Protocol*) dan UDP (*User Datagram Protocol*). TCP bersifat *connection oriented* sedangkan UDP bersifat *connectionless*.

2.1.3.5 Lapisan Application

Lapisan inilah biasa disebut lapisan akhir (*front end*) atau biasanya disebut *user program* dimana merupakan sebuah aplikasi yang mengirimkan data ke lapisan *transport*. Misalnya FTP, email programs dan *web browsers*.

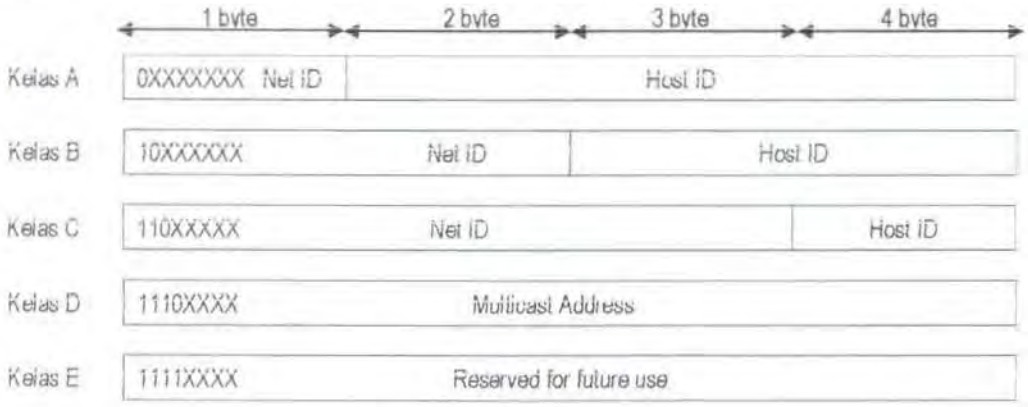
2.2 Pengalamatan IP

Untuk membantu mencapai komputer tujuan, setiap komputer dalam jaringan TCP/IP harus diberikan minimal sebuah alamat IP. Alamat tersebut terdiri

atas 32 bit biner yang bersifat unik dan berlaku secara umum yang mendefinisikan *host* atau router di internet. 32 bit biner tersebut terbagi menjadi empat bagian yang mempunyai nilai dari 0-255 yang sering ditulis dalam bentuk [xxx.xxx.xxx.xxx]. Alamat ini mendefinisikan dua bagian yaitu ***net ID*** dan ***host ID***. *Net ID* mendefinisikan sebuah jaringan dan *host ID* mendefinisikan sebuah *host* pada jaringan tersebut.

2.2.1 Pembagian Kelas

Ada lima kelas yang berbeda pada alamat IP yaitu kelas A, B, C, D dan E. Kelas ini didesain untuk mencukupi kebutuhan akan jenis organisasi yang berbeda



Gambar 2.2 Kelas Alamat IP

Berikut keterangan masing-masing kelas alamat IP:

2.2.1.1 Kelas A

Pada kelas A, delapan bit yang pertama mendefinisikan *net ID*. Tetapi bit yang paling kiri harus 0(nol) yang berfungsi untuk mendefinisikan sebagai kelas A. Tujuh bit yang tersisa mendefinisikan jaringan yang berbeda. Secara teori akan

terdapat $2^7 = 128$ jaringan. Namun sebenarnya ada 126 jaringan di kelas A karena dua alamat digunakan sebagai cadangan untuk tujuan tertentu. *Host ID* didefinisikan oleh 24 bit. Artinya secara teori masing-masing jaringan memiliki $2^{24} = 16.777.216$ *host*. Namun seperti pada *net ID*, dua alamat khusus (*host ID* yang semua 0 dan yang semua 1) digunakan untuk alamat khusus.

2.2.1.2 Kelas B

Pada kelas B 16 bit mendefinisikan *net ID* dan 16 bit lain mendefinisikan *host ID*. Dua bit yang paling kiri adalah 10 yang berfungsi untuk mendefinisikan sebagai kelas B. 14 bit selanjutnya mendefiniskan jaringan yang berbeda sehingga jumlah jaringan adalah $2^{14} = 16.384$ jaringan. Jadi jumlah jaringan yang lebih banyak di kelas B dibandingkan di kelas A. *Host ID* didefinisikan oleh 16 bit sehingga masing-masing jaringan memiliki $2^{16} = 65.536$ *host* atau router. Namun dua alamat (yang semua 0 dan semua 1) digunakan untuk alamat khusus.

2.2.1.3 Kelas C

Pada kelas C 24 bit mendefinisikan *net ID* dan 8 bit mendefinisikan *host ID*. Tiga bit yang paling kiri adalah 110 yang berfungsi untuk mendefinisikan sebagai kelas C. 21 bit selanjutnya mendefiniskan jaringan yang berbeda sehingga jumlah jaringan ada $2^{21} = 2.97.152$ jaringan. Jadi jumlah jaringan yang lebih banyak di kelas C dibandingkan di kelas A dan B. *Host ID* didefinisikan oleh 8 bit. Masing-masing jaringan memiliki $2^8 = 256$ *host* atau router. Namun dua alamat (yang semua 0 dan semua 1) digunakan untuk alamat khusus.

2.2.1.4 Kelas D

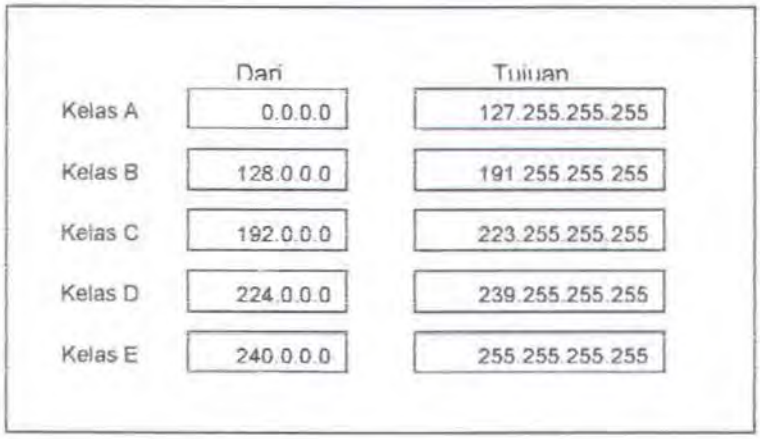
Alamat pada kelas D digunakan untuk *multicast*ing. Pada kelas ini, tidak ada *net ID* maupun *host ID*. Semua alamat digunakan untuk *multicasting*. Empat bit yang pertama mendefinisikan kelas D yaitu 1110. 28 bit selanjutnya mendefinisikan alamat *multicast* yang berbeda.

2.2.1.5 Kelas E

Kelas E digunakan sebagai cadangan internet untuk kegunaan khusus dan eksperimen. Tidak ada *net ID* dan *host ID* pada kelas ini. Empat bit pertama mendefinisikan kelas E yaitu 1111.

2.2.2 Penentuan Kelas IP Address

Untuk menentukan kelas dari alamat IP bisa dilihat dari alamat IP tersebut jika yang diketahui adalah bentuk biner dengan melihat bit pertama sesuai dengan kelas yang telah didefinisikan diatas. Jika alamat yang didefinisikan adalah bentuk desimal, maka hanya perlu melihat angka pertama untuk mendefinisikan kelas. (lihat gambar 2.3)



Gambar 2. 3 Kelas dengan notasi decimal

2.2.3 Alamat Khusus

Beberapa bagian dari alamat di kelas A, B dan C digunakan untuk alamat khusus.

Tabel 2. 1 Alamat Khusus

Special Address	Net ID	Host ID	Source or Destination
Nerwork Address	Specific	All 0s	None
Direct broadcast address	Specific	All 1s	Destination
Limited broadcast address	All 1s	All 1s	Destination
This host on this network	All 0s	All 0s	Source
Specific host on this network	All 0s	Specific	Destination
Loopback address	127	Any	Destination

Berikut ini keterangan masing-masing dari alamat khusus :

2.2.3.1 Network Address

Pada kelas A, B, dan C sebuah alamat dengan *host ID* semuanya yang terdiri dari 0 tidak diberikan ke *host* manapun tapi digunakan sebagai cadangan untuk mendefinisikan alamat jaringannya sendiri. Alamat ini dinamakan *Network Address*. Alamat ini tidak bisa untuk mendefinisikan alamat sumber atau tujuan pada paket IP. Alamat khusus ini juga mengurangi jumlah ketersediaan *host ID* untuk masing-masing *net ID* pada kelas A, B dan C.

2.2.3.2 Direct Broadcast Address

Pada kelas A, B dan C, jika *host ID* adalah semuanya 1 maka alamat ini disebut *direct broadcast address*. Alamat ini digunakan oleh router untuk mengirim sebuah paket ke semua *host* pada jaringan tertentu. Semua *host* akan

menerima sebuah paket yang memiliki jenis alamat tujuan yang sama. Alamat ini hanya bisa digunakan sebagai alamat tujuan pada paket IP. Alamat ini juga mengurangi jumlah ketersediaan *host ID* untuk masing-masing *net ID* pada kelas A, B dan C.

2.2.3.3 Limited Broadcast Address

Pada kelas A, B dan C, sebuah alamat dengan *net ID* dan *host ID* (32 bit) yang diberi nilai 1 semua digunakan untuk mendefinisikan alamat *broadcast* pada jaringan saat itu. Sebuah *host* yang akan mengirim pesan untuk setiap *host* yang lain dapat menggunakan alamat ini sebagai alamat tujuan pada paket IP. Tetapi router akan menahan paket yang memiliki jenis alamat ini untuk menahan *broadcasting* ke jaringan lokal. Alamat ini terdapat pada kelas E.

2.2.3.4 This Host on This Network

Jika alamat IP terdiri dari angka 0 semua, alamat ini dinamakan *this host on this network*. Alamat ini digunakan oleh *host* pada *bootstrap time* ketika tidak mengetahui alamat IP-nya. *Host* mengirim paket IP ke *bootstrap* server menggunakan alamat ini sebagai alamat sumber dan sebuah *limited broadcast address* sebagai alamat tujuan yang berfungsi untuk mendapatkan alamatnya sendiri. Alamat ini hanya bisa digunakan sebagai alamat sumber dan berada pada kelas A tanpa memperhatikan jaringan.

2.2.3.5 Specific Host on This Network

Sebuah alamat IP dengan *net ID* yang semuanya 0 dinamakan *specific host on this network*. Alamat ini digunakan oleh *host* untuk mengirim pesan ke *host*

yang lain pada jaringan yang sama. Karena paket diblok oleh router, maka alamat ini digunakan sebagai jalan untuk menahan paket ke jaringan lokal. Alamat ini hanya digunakan untuk alamat tujuan dan berada pada kelas A tanpa memperhatikan jaringannya.

2.2.3.6 Loopback Address

Alamat IP dengan byte pertama sama dengan 127 digunakan untuk *loopback address*, merupakan alamat yang digunakan untuk mengecek software pada mesin dan juga untuk melakukan tes IP software. Ketika alamat ini digunakan, sebuah paket tidak pernah meninggalkan mesin dan dengan mudah kembali ke protokol software. *Loopback address* dapat digunakan oleh proses klien untuk mengirim pesan ke proses server pada mesin yang sama. Alamat ini hanya digunakan sebagai alamat tujuan pada paket IP dan berada pada kelas A.

2.2.4 Private Network IP

Alamat *Private Network* IP ini hanya digunakan untuk jaringan pribadi, jadi alamat alamat IP ini tidak digunakan dalam Internet. Alamat yang juga disebut *virtual address* ini dapat digunakan pada kelas A, B dan C tanpa melakukan pendaftaran ke *Internet Authorities*. Alamat *Private Network* IP pada tiap-tiap kelas terapat pada tabel 2.2.

Tabel 2. 2 Alamat Private

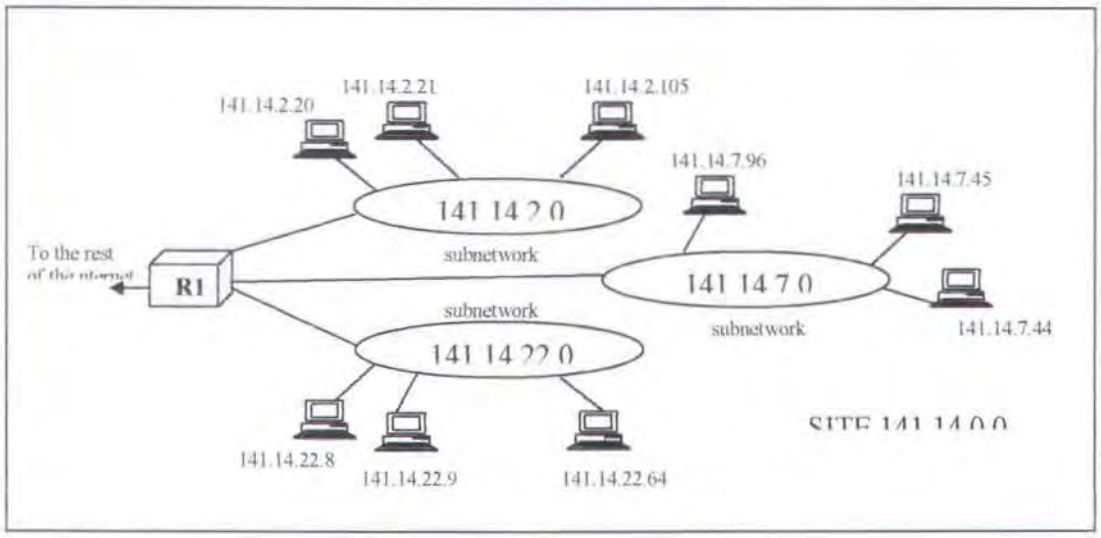
Kelas	Alamat	Jumlah
A	10 . x . x . x	1
B	172 . 16 . x . x – 172 . 31 . x . x	16
C	192 . 168 . 0 . 0 – 192 . 168 . 255 . 255	256

2.2.5 Subnetting

Pada pengalamatan IP terdapat bagian yang menunjukkan *net ID* dan bagian yang menunjukkan *host ID*. Untuk menuju suatu *host* di internet, maka kita harus mencapai jaringan dengan menggunakan alamat *net ID* dan kemudian baru menuju ke *host* dengan menggunakan alamat *host ID*. Dengan kata lain, kelas A, B dan C dalam pengalamatan IP didesain menjadi dua *level* hirarki.

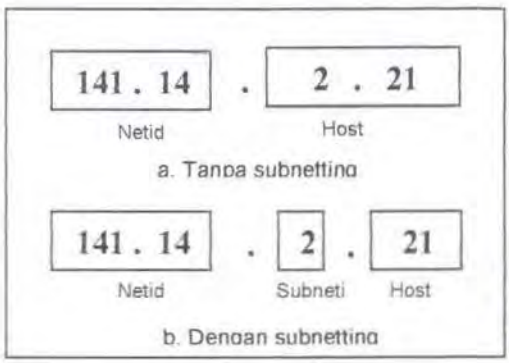
Namun pada beberapa kasus, dua *level* hirarki masih belum cukup. Maka untuk mengatasinya adalah dengan menggunakan subnetting, yaitu pembagian network menjadi beberapa network yang lebih kecil yang disebut subnetwork. Pada gambar 2.5 terlihat perbedaan networking yang memakai subnetting dan tanpa subnetting.

Dari gambar 2.4, pada awalnya internet tidak mengetahui bahwa jaringan telah dibagi menjadi tiga subnetwork. Sebuah paket yang beralamatkan untuk *host* 141.14.2.21 masih mencapai router R1. Alamat tujuan dari *datagram* IP masih berupa alamat kelas B dimana 141.14 mendefinisikan *net ID* dan 2.21 mendefinisikan *host ID*. Ketika *datagram* mencapai router R1, interpretasi dari alamat IP berubah. Router R1 mengetahui bahwa jaringan 141.14 secara fisik dibagi menjadi 3 subnetwork. Maka 2.21 harus diinterpretasikan sebagai *subnet ID* 2 dan *host ID* 21. Router R1 menggunakan dua byte pertama (141.14) sebagai *net ID*, byte ketiga (2) sebagai *subnet ID* dan byte keempat (21) sebagai *host ID*.



Gambar 2. 4 Network dengan 3 level hirarki (subnet)

Penambahan subnetwork mengakibatkan adanya penambahan *level* hirarki pada sistem pengalamatan IP. Ada tiga *level* hirarki yaitu *net ID*, *subnet ID* dan *host ID*. *Net ID* adalah *level* pertama yang mendefinisikan *site*, *level* kedua adalah *subnet ID* yang mendefinisikan subnetwork fisik dan *host ID* adalah *level* ketiga yang mendefinisikan koneksi dari *host* ke subnetwork.

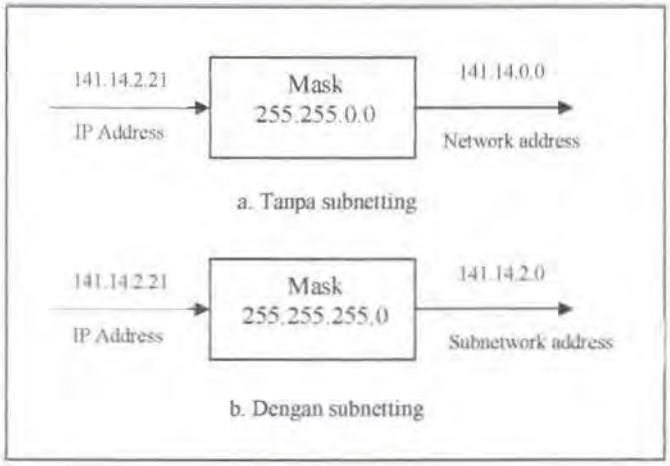


Gambar 2. 5 Pengalamatan Jaringan dengan dan tanpa subnetting

2.2.6 Masking

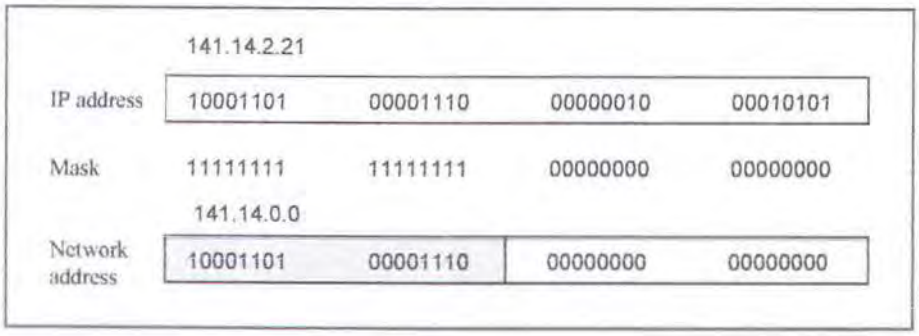
Masking adalah proses mengekstrak alamat jaringan fisik dari alamat IP. *Masking* dapat dilakukan tanpa tergantung apakah kita memiliki *subnetting* atau tidak. Jika kita tidak menggunakan *subnetting*, maka *masking* dilakukan dengan cara mengekstrak alamat jaringan dari suatu alamat IP. Namun jika kita menggunakan *subnetting*, maka *masking* dilakukan dengan mengekstrak alamat subnetwork dari alamat IP (lihat gambar 2.6).

Dalam *masking*, kita menggunakan operasi matematika 32 bit alamat IP pada *level* bit menggunakan 32 bit yang lain yang disebut *mask*. Bit pada *mask* berhubungan dengan bit pada alamat IP. Bagian dari *mask* yang terdiri dari angka 1 mendefinisikan *net ID* atau kombinasi *net ID* dan *subnet ID*. Bagian dari *mask* yang terdiri dari angka 0 mendefinisikan *host ID*. Untuk mencapai jaringan atau alamat subnet harus dilakukan operasi *bit-wise-and* pada alamat IP dan *mask*. Hasil dari operasi tersebut akan didapatkan alamat network.

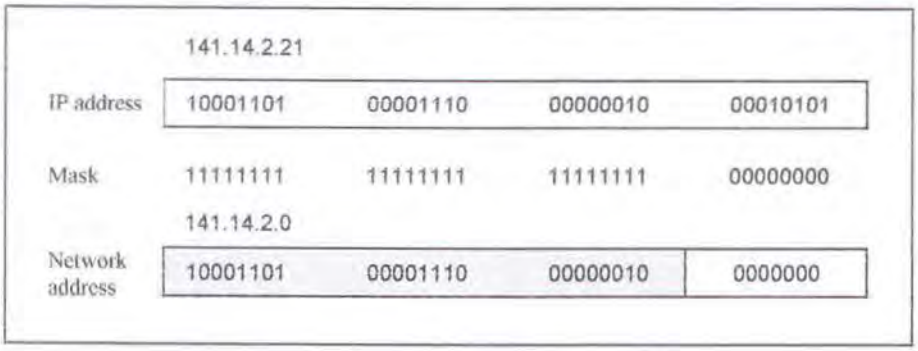


Gambar 2. 6 Masking

Gambar 2.7 menunjukkan bagaimana bagaimana cara menggunakan *mask* baik yang menggunakan *subnetting* maupun tidak. Nilai network *address* akan berbeda antara yang menggunakan subnetting dan yang tidak menggunakannya hal ini disebabkan karena nilai *mask*-nya juga berbeda.



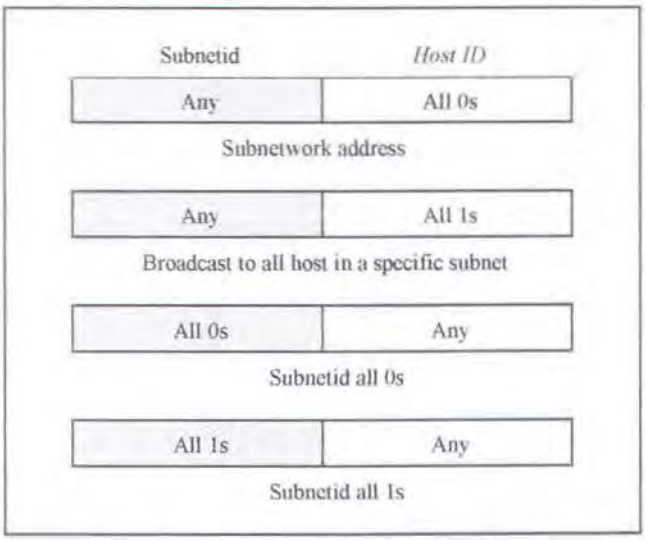
a. Tanpa subnetting



b. Dengan subnetting

Gambar 2. 7 Penggunaan Masking

Pada subnetting, beberapa alamat digunakan secara khusus(lihat gambar 2.8). *Subnet ID* yang semuanya 1 atau 0 tidak digunakan oleh *host* manapun. Alamat dengan *host ID* yang semuanya 1 digunakan untuk *broadcast*ing ke semua *host* pada subnet khusus. Alamat dengan *host ID* yang semuanya 0 juga digunakan untuk mendefinisikan subnetwork itu sendiri.



Gambar 2. 8 Alamat khusus pada subnetting

2.2.7 Alamat Komunikasi

Komunikasi pada internet dapat terjadi dengan menggunakan alamat *unicast*, *multicast* atau *broadcast*.

2.2.7.1 Alamat Unicast

Komunikasi *unicast* adalah *one-to-one*. Ketika paket dikirim dari sumber individual ke tujuan individual, maka komunikasi *unicast* terjadi. Alamat *unicast* ada pada kelas A, B atau C dan sifatnya unik.

2.2.7.2 Alamat Multicast

Komunikasi *multicast* adalah *one-to-many*. Ketika paket dikirim dari sumber individual ke beberapa tujuan, maka komunikasi *multicast* terjadi. Pengiriman paket pada alamat *multicast* hanya akan diterima oleh beberapa alamat yang tergabung dalam satu kelompok alamat *multicast* tersebut. Alamat *multicast* adalah alamat kelas D yang hanya digunakan sebagai alamat tujuan.

Sebuah sistem di internet bisa memiliki satu atau lebih alamat *multicast* pada kelas D.

Internet Authorities telah memberikan beberapa alamat *multicast* untuk beberapa kelompok khusus:

a. Kategori

Alamat *multicast* ini dimulai dengan awalan 224.0.0 dan digunakan untuk tujuan khusus.

b. Konferensi

Digunakan untuk konferensi dan telekonferensi. Alamat *multicast* ini dimulai dengan awalan 224.0.1

2.2.7.3 Alamat Broadcast

Komunikasi *broadcast* adalah *one-to-all. Broadcast*ing hanya dapat terjadi pada *level* lokal. Tidak ada *broadcasting* pada *level* global, artinya sebuah sistem tidak bisa mengirim pesan ke semua *host* dan router di internet. Hal ini terjadi untuk mencegah terjadinya kemacetan lalu lintas.

2.3 ARP

2.3.1 Definisi ARP

Pada pemetaan dinamis setiap waktu sebuah mesin mengetahui satu dari dua alamat *logical* maupun fisik. Alamat logikal pada protokol TCP/IP adalah alamat IP, sedangkan alamat fisik merupakan alamat hardware. Untuk menemukan sebuah alamat fisik dari alamat *logical* atau sebaliknya dapat digunakan protokol. Dua protokol telah didesain untuk menyajikan pemetaan

dinamis, yaitu *Address Resolution Protocol* (ARP) dan *Reverse Address Resolution Protocol* (RARP). Pemetaan pertama merupakan pemetaan dari alamat *logical* ke alamat fisik. Pemetaan kedua merupakan pemetaan dari alamat fisik menuju alamat *logical*.

ARP dan RARP menggunakan alamat fisik *unicast* dan *broadcast*. Dalam hal ini berarti sebagai contoh Ethernet menggunakan seluruh alamat 1 (FFFFFFFFF16) sebagai alamat *broadcast*.

Setiap kali sebuah *host* atau router memiliki *datagram* IP untuk dikirimkan ke *host* atau router lain terdapat alamat *logical* (IP) dari penerima. Tapi *datagram* IP harus dilindungi pada bingkai agar dapat melewati jaringan fisik. Ini berarti bahwa pengirim membutuhkan alamat fisik dari penerima. Pemetaan menghubungkan alamat *logical* menuju alamat fisik.

Hal ini dapat dilakukan secara statis maupun dinamis. Hubungan antara alamat logical dan fisik dapat disimpan secara statis pada tabel. Pengirim dapat melihat pada tabel dan mencari alamat fisik yang berhubungan dengan alamat logik. Hal ini bukanlah solusi yang baik mengingat bahwa setiap saat alamat fisik berubah sehingga tabelnya harus diubah pula. Perubahan ini dapat pula dilakukan secara dinamis dimana pengirim meminta penerima untuk memberitahu alamat fisik ketika diperlukan. ARP didesain untuk tujuan ini.

ARP berhubungan dengan alamat IP dengan alamat fisiknya. Pada jaringan fisik tertentu seperti LAN, tiap perangkat pada hubungan diidentifikasikan oleh alamat fisik ataupun alamat stasiun yang biasa dikeluarkan oleh NIC.

Kapanpun *host* dan router membutuhkan alamat fisik maupun *host* atau router pada jaringannya, dikirimkanlah paket pencarian ARP. Paket termasuk alamat fisik dan IP dari pengirim dan alamat IP dari penerima. Karena pengirim tidak mengetahui alamat fisik dari penerima, pencarian dilakukan dengan *broadcast* ke seluruh jaringan tersebut.

2.3.2 Paket ARP

Sebuah paket ARP mempunyai format protokol seperti gambar 2.9 di bawah ini :

<table>
<tr><td colspan="2">Hardware Type</td><td>Protocol Type</td></tr>
<tr><td>Hardware Length</td><td>Protocol Length</td><td>Operation
Request 1, Reply 2</td></tr>
<tr><td colspan="3">Sender Hardware Address (misalnya 6 bytes untuk ethernet)</td></tr>
<tr><td colspan="3">Sender Protocol Address (misalnya 4 bytes untuk IP)</td></tr>
<tr><td colspan="3">Target Hardware Address (misalnya 6 bytes untuk ethernet)</td></tr>
<tr><td colspan="3">Target Protocol Address (misalnya 4 bytes untuk IP)</td></tr>
</table>

Gambar 2. 9 Paket ARP

Keterangan masing-masing field :

1. Hardware Type (HTYPE)

   Terdiri atas 16 bit mendefinisikan jenis network dimana ARP berjalan. Setiap LAN mempunyai nilai integer sendiri, misalnya Ethernet bernilai 1.

2. Protocol Type (PTYPE)

   Terdiri atas 16 bit mendefinisikan jenis protokol. Misalnya IPv4 bernilai 0800_{16}.

3. Hardware Length (HLEN)

Terdiri atas 8 bit berisi panjang alamat fisik dalam bentuk byte, misalnya Ethernet bernilai 6.

4. Protocol Length (PLEN)

   Terdiri atas 8 bit berisi panjang alamat fisik dalam bentuk byte, misalnya Ethernet bernilai 6.

5. Operation (OPER)

   Terdiri atas 16 bit berisi jenis dari paket. Ada dua jenis paket yaitu ARP *Request* dan ARP *Reply*.

6. Sender Hardware Address (SHA)

   Panjang field ini bervariasi. Field ini mendefinisikan alamat fisik dari pengirim. Misalnya Ethernet memiliki panjang 6 byte.

7. Sender Protocol Address (SPA)

   Panjang field ini bervariasi. Field ini mendefinisikan alamat *logical* dari pengirim. Misalnya IP memiliki panjang 4 byte.

8. Target Hardware Address (THA)

   Panjang field ini bervariasi. Field ini mendefinisikan alamat fisik tujuan atau penerima. Misalnya Ethernet memiliki panjang 6 byte.

9. Target Protocol Address (TPA)

   Panjang field ini bervariasi. Field ini mendefinisikan alamat *logical* tujuan atau penerima. Misalnya IP memiliki panjang 4 byte.

2.3.3 Komponen ARP

Paket ARP melibatkan lima komponen yaitu *Cache table*, *Queue*, Modul Output, Modul Input dan Modul *Cache*. Di bawah ini keterangan masing-masing

komponen :

2.3.3.1 Cache table

Pengirim mempunyai lebih dari satu *datagram* IP untuk dikirimkan ke tujuan yang sama. Penggunaan protokol ARP untuk tiap *datagram* ditentukan untuk *host* yang sama atau router tidaklah efisien[1]. Hal ini disebabkan penggunaan *cache table* yang berguna untuk menyimpan alamat fisik yang bersesuaian untuk *datagram* IP ketika *host* atau router menerima alamat fisik. Padahal ruang pada *cache table* sangat terbatas dan pemetaan pada *cache* tidak boleh dihasilkan dalam waktu tidak terbatas.

Cache table diimplementasikan sebagai pemasukan *array*. Pada desain ini, tiap masukan terdiri dari *field* berikut ini:

- *State. Field* ini menunjukkan posisi dari masukan. Nilainya bisa berupa FREE, PENDING atau RESOLVED. Status FREE berarti bahwa waktu untuk hidup dari pemasukan ini telah berakhir. Ruang dapat digunakan untuk masukan baru. Status PENDING berarti permintaan atas masukan ini telah dikirimkan, namun balasan belum diterima. Sedangkan status RESOLVED berarti pemasukan telah selesai. Paket yang menunggu untuk dikirimkan ke tujuan dapat menggunakan informasi ini dalam pemasukan.
- *Hardware type. Field* ini bernilai sama dengan field paket ARP.
- *Protocol type. Field* ini bernilai sama dengan field paket ARP.
- *Hardware length. Field* ini bernilai sama dengan field paket ARP.
- *Protocol length. Field* ini bernilai sama dengan field paket ARP.

- *Interface number.* Sebuah router atau *multihomed host* dapat dihubungkan ke jaringan yang berbeda dengan nomer *interface* yang berbeda.
- *Queue number.* ARP menggunakan nomor antrian yang berbeda untuk menyalurkan paket yang menunggu masukan ini untuk diputuskan.
- *Attemps. Field* ini menunjukkan berapa kali permintaan ARP dikirimkan untuk pemasukan ini.
- *Time-out. Field* ini menunjukkan waktu hidup dari pemasukan dalam hitungan detik.
- *Hardware address. Field* ini menunjukkan alamat hardware tujuan yang akan kosong sampai ada balasan dari ARP.
- *Protocol address. Field ini menunjukkan alamat IP tujuan.*

2.3.3.2 Modul Output

Modul output menunggu sebuah paket IP dari perangkat lunak IP. Modul ini memeriksa *cache table* untuk menemukan pemasukan yang sesuai dengan alamat IP dari paket ini. IP tujuan dari paket IP harus cocok dengan alamat protokol dari pemasukan.

Bila masukan ditemukan dan status dari masukan adalah RESOLVED, paket sepanjang alamat hardware tujuan dilewatkan lapisan *data link* untuk transmisi. Bila status dari masukan PENDING, paket harus menunggu sampai alamat hardware tujuan ditemukan. Sebuah antrian dibuat untuk tujuan ini. Setelah itu modulnya mengirimkan paket ke antrian yang sesuai.

Bila tidak ada masukan yang ditemukan, modul membuat antrian dan mengantrikan paket. Masukan baru dengan status PENDING dibuat untuk tujuan

ini dan nilai dari *field* ATTEMPTS diatur menjadi nilai 1. Paket Permintaan ARP kemudian menjadi *broadcast*.

2.3.3.3 Modul Input

Modul input menunggu sampai paket ARP (*request* atau *reply*) datang. Modul input memeriksa *cache table* untuk menemukan masukan yang berhubungan dengan paket ARP. Tujuan alamat protocol harus sesuai dengan alamat yang dimasukkan.

2.3.3.4 Queues

Pada desain ini, paket ARP menangani kumpulan dari antrian, dimana untuk tiap tujuan untuk menahan paket IP selama ARP berusaha untuk memecahkan alamat hardware. Modul output mengirimkan paket yang tidak terselesaikan pada antrian bersesuaian. Modul input menggantikan paket dari antrian dan mengirimkannya dengan alamat fisik yang telah terselesaikan ke lapisan *data link* untuk transmisi.

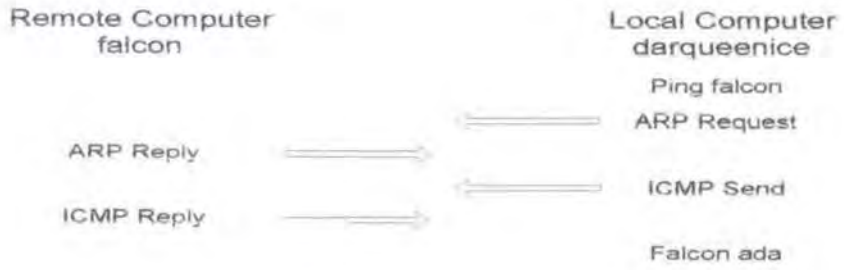
2.3.3.5 Cache-Control

Modul *cache control* bertanggung jawab untuk memelihara *cache table*. Modul ini secara periodik memeriksa *cache tabel* setiap inputnya.

2.3.4 Bagimana ARP Bekerja

Ketika datang sebuah paket yang dituju untuk sebuah *host* pada sebagian LAN di *gateway*, *gateway* menjalankan program ARP untuk mencari Alamat MAC yang cocok dengan IP address. Program ARP merujuk pada ARP cache dan

jika mendapat alamat tersebut, paket tersebut dapat diubah sesuai panjang dan format paket dan dikirim ke *host*. Jika tidak mendapatkan, ARP akan *broadcast* sebuah paket *request* dalam format khusus ke seluruh *host* dalam LAN tersebut untuk melihat jika sebuah *host* mempunyai kesesuaian dengan IP address yang dimaksud. Sebuah *host* yang mengenali IP address itu adalah IP miliknya akan melakukan reply sebagai tanda. ARP akan memperbaharui ARP cache sebagai referensi yang akan datang dan mengirimkan paket ke alamat MAC yang melakukan reply tersebut. Gambar 2.10 contoh bagaimana ARP bekerja dalam proses ping.



Gambar 2. 10 Cara Kerja ARP

Jadi sebelum paket ICMP dikirim, terlebih dahulu dikirimkan pesan ARP. Jika kita memakai program tcpdump maka akan terlihat seperti gambar 2.11 dan balasannya seperti pada gambar 2.12.

```
12:00:00.100028 arp who-has falcon tell darqueenice
```

Gambar 2. 11 Capture TCPDump Request

```
12:02:00.100028 arp reply falcon is at 00:51:00:10:00:28
```

Gambar 2. 12 Capture TCPDump Reply

2.4 MAC

2.4.1 Definisi MAC

Alamat *Media Control Access* (MAC) biasanya disebut sebagai alamat hardware merupakan alamat fisik untuk piranti yang terhubung ke jaringan. Alamat MAC adalah bagian dari lapisan *data link* yang berfungsi untuk mendeteksi *error*, membagi data menjadi banyak *frame*, menambahkan *header* pada *frame* dan memastikan bahwa data telah diterima dengan baik. Alamat ini diberikan kepada kepada semua peralatan yang digunakan pada jaringan seperti *Net Interface Card* (NIC), *Interface Router*, *Switch* dan *Hub*.

2.4.2 Format MAC

Sebuah alamat MAC mempunyai nilai unik yang bersesuaian dengan *network adapter*. Alamat MAC mempunyai 12 digit angka hexadesimal (panjangnya 48 bit) dimana biasanya ditulis dengan format di bawah ini:

```
MM:MM:MM:SS:SS:SS
```

Enam digit pertama melambangkan nomor seri dari pabrik yang membuat. ID ini merupakan standar internet. Sedangkan enam digit kedua melambangkan nomor seri. Contoh :

```
00:A0:C9:14:C8:29
```

Nilai awal 00A0C9 menyatakan ID Intel Corporation. Sedangkan nilai 14C829 menyatakan nomor seri.

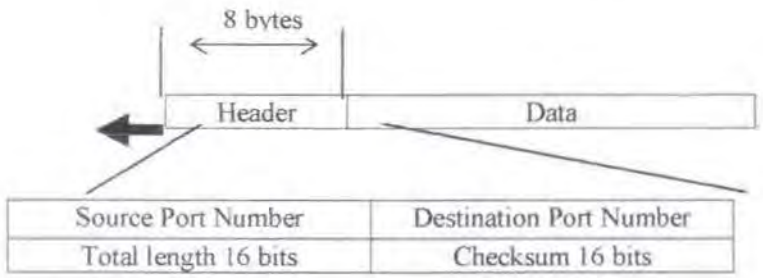
2.5 User Datagram Protocol (UDP)

UDP merupakan protokol bersifat *connectionless* terdapat pada lapisan *transport* yang berada diantara lapisan *application* dan lapisan *network* yang merupakan perantara aplikasi program dan operasi jaringan. UDP tidak memiliki tambahan pada servis IP kecuali menyediakan komunikasi *process to prosess* daripada komunikasi *host to host*. UDP juga menyediakan *cheking error* yang sangat terbatas. UDP tidak mengurus masalah penerimaan aliran data dan pembuatan segmen yg sesuai untuk IP. Akibatnya, kemampuan UDP berada dibawah TCP.

Dengan beberapa kerugian yang ada pada UDP ada beberapa keuntungan yang menjadikan UDP digunakan dalam sebuah proses yang menginginkan pengiriman pesan dengan ukuran kecil dan tidak terlalu memperhatikan reliabilitas, selain itu UDP dapat digunakan untuk komnukiasi yang membutuhkan interaksi antara pengirim dan penerima yang lebih sedikit dibanding dengan TCP.

2.5.1 Paket UDP

Paket UDP yang disebut *user datagrams* memiliki ukuran *header* yang ditentukan yaitu 8 byte. Format UDP terdapat pada gambar 2.13.



Gambar 2. 13 Format User datagram

- Source port number

  Nomor port ini digunakan oleh proses yang berjalan di *host* sumber dan memiliki panjang 16 bit sehingga memiliki nomor port antara 0 sampai dengan 65535.

- Destination port number

  Port number yang digunakan oleh proses yang berjalan pada *host* tujuan dan juga memiliki panjang 16 bit

- Length

  Mendefinisikan panjang total user *datagram*, header dan data.

- Checksum

  Digunakan untuk mendeteksi error seluruh user *datagram* (header dan data)

2.5.2 Kegunan UDP

Berikut ini beberapa kegunaan UDP:

- UDP cocok untuk proses yang membutuhkan komunikasi sederhana dan sedikit kontrol aliran dan kesalahan. Oleh karena itu UDP biasanya tidak digunakan pada proses pengiriman data dalam jumlah besar seperti FTP.
- UDP cocok untuk proses dengan aliran internal dan mekanisme kontrol kesalahan seperti *Trivial File Transfer Protocol* (TFTP).
- UDP cocok untuk transportasi protokol *multicast*ing dan *broadcast*ing.
- UDP digunakan untuk pengaturan proses seperti SNMP.

UDP digunakan untuk pembahuruan *routing* seperti *Routing Information Protocol* (RIP).

2.6 Socket

Istilah *socket* sebenarnya mengacu pada *Application Program Interface* (API) yang dimplementasikan oleh *Berkeley Software Distribution* (BSD) UNIX pada tahun 1983. *Socket* pertama dikembangkan oleh Universitas California pada Berkeley dan disertakan pada versi 4.1, 4.2 dan 4.3 dari sistem BSD UNIX.

Karena BSD UNIX telah dijalankan pada banyak arsitektur mesin untuk *dekstop workstation*, untuk *mainframe* besar dan berbagai aplikasi banyak menggunakan BSD *socket* sebagai penghubung komunikasi jaringan. Dengan adanya bahasa C sekarang *socket* juga dapat digunakan pada sistem IBM dan semakin banyak lagi aplikasi yang berjalan di bawah sistem MVS. Sebuah library fungsi *socket* BSD dapat menyederhanakan servis komunikasi yang dibutuhkan aplikasi.

Ketika suatu program melakukan proses input atau output maka program tersebut akan membaca atau menulis pada file deskriptor. *Socket* merupakan salah satu file deskriptor untuk networking. File deskriptor hanya sebuah integer yang berasosiasi dengan suatu piranti, terminal, suatu koneksi pada network, atau yang lainnya. File deskriptor ini dapat dipandang sebagai komunikasi antar programmer dengan system. Dengan *socket* maka komunikasi dapat dilakukan dengan membuka *socket*, menulis lalu menutup setelah *socket* tidak digunakan. Sebagai contoh, suatu printer port hardware berasosiasi dengan file */dev/lpt1*, maka untuk memberikan informasi atau mengambil informasi dari port tersebut yang kita lakukan adalah membuka akses ke file tersebut dan membacanya atau menuliskan informasi.

2.7 IOCTL

IOCTL adalah sebuah fungsi yang menyediakan berbagai kegunaan pada jaringan seperti mengubah karakter sebuah *socket*, *routing table*, table ARP dan karakteristik *interface*. Deklarasi fungsi IOCTL terdapat pada gambar 2.14.

```
int ioctl(int, int, ...)
```

Gambar 2. 14 Deklarasi IOCTL

Argumen pertama adalah sebuah file deskriptor. Karena semua piranti di LINUX diakses seperti file. File deskriptor biasanya digunakan untuk membuka *device* yang kita ingin gunakan.

Argumen kedua pada fungsi ioctl() adalah sebuah integer yang menyatakan nomor khusus tergantung pada pirantinya. Sebagai contoh kita ingin menambah kecepatan *device* serial bukan printer tentunya nomor yang diberikan adalah nomor serial.

Argumen tambahan tidak harus diisi hal ini tergantung dari implementasi ioctl itu terhadap *device* tertentu. Biasanya argumen ketiga merupakan *pointer* dari sebuah *struct* dimana data-data yang terdapat dalam *struct* dibutuhkan untuk berhubungan dengan *device* tersebut.

Sebagai contoh bagaimana ioctl bekerja di bawah ini adalah contoh program sederhana untuk memeriksa salah satu *signal* dari port serial. Program pada gambar 2.15 membuka sebuah tty dari serial port dan memanggil ioctl dengan file deskriptor dari port serial. Argumen **TIOCMGET** berfungsi untuk melihat status bit modem.

```
#include <termios.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/ioctl.h>

main()
{
  int fd, status;
   fd = open("/dev/ttyS0", O_RDONLY);
   if (ioctl(fd, TIOCMGET, &status) == -1)
      printf("TIOCMGET failed: %s", strerror(errno));
   else {
      if (status & TIOCM_DTR)
         puts("TIOCM_DTR is not set");
      else
         puts("TIOCM_DTR is set");
   }
   close(fd);
}
```

Gambar 2. 15 Program Port Serial

Penggunaan ioctl pada Ethernet *driver* mempunyai proses yang sama. Argumen ketiga dari ioctl adalah *pointer* dari *struct ifreq*. Deklarasi dari ifreq terdapat pada net/if.h. Berikut isi dari *struct ifreq* pada gambar 2.16.

Pada *struct ifreq* memiliki lima variabel yang bertipe data *sockaddr* yaitu *ifru_addr*, *ifru_dstaddr*, *ifru_broadaddr*, *ifru_netmask* dan *ifru_hwaddr*. *Ifru_addr* mengandung alamat IP dari interface tersebut. *ifru_dstaddr* mengandung alamat tujuan jika interface tersebut melakukan *point to point* dengan *interface* lain. *Ifru_broaddr* mengandung alamat *broadcast interface*. *Ifru_netmask* mengandung nilai *netmask* yang dipakai oleh *subnet interface*. *Ifru_hwaddr* mengandung alamat MAC atau *hardware* dari interface tersebut.

```
struct ifreq
{
#define IFHWADDRLEN     6
#define     IFNAMSIZ    16
      union
      {
      char ifrn_name[IFNAMSIZ]; // nama interace
      } ifr_ifrn;
            union {
            struct sockaddr ifru_addr; // IP Address
            struct sockaddr ifru_dstaddr; //p_p link lain
            struct sockaddr ifru_broadaddr; // Broadcast
Address
            struct sockaddr ifru_netmask; // Netmask
            struct sockaddr ifru_hwaddr; // MAC Address
            short ifru_flags;
            int ifru_ivalue;
            int ifru_mtu;
            struct ifmap ifru_map;
            char ifru_slave[IFNAMSIZ];
            char ifru_newname[IFNAMSIZ];
            char * ifru_data;
            struct if_settings ifru_settings;
      } ifr_ifru;
};
```

Gambar 2. 16 Struct Ifreq

2.8 Cyclic Redudancy Check (CRC

Ketika sebuah data terkirim melalui media yang dapat mengalami kesalahan sehingga diperlukan mekanisme pengecekkan kesalahan. Salah satu mekanisme tersebut adalah *Cyclic Redundancy Check* (CRC). CRC menghitung blok data dengan membaca blok data sebagai string yang diambil dari nilai biner

masing-masing karakter dari string. Contoh:

Bentuk string	1	2	3
Bentuk desimal	48	49	50
Bentuk hexadesimal	30	31	32
Bentuk biner	00110000	00110001	00110010

Bentuk biner diatas itulah yang akan diproses dalam CRC.)

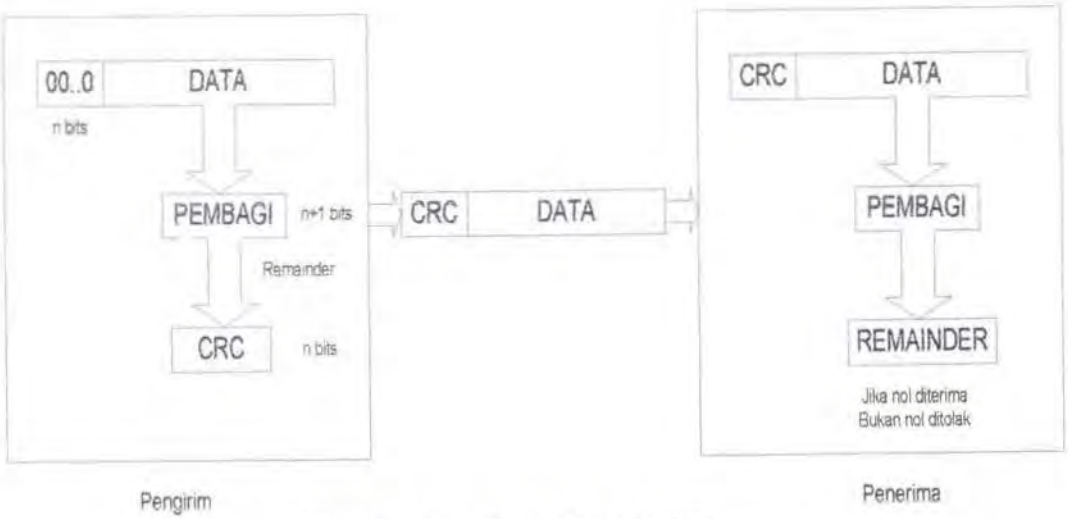
2.8.1 Mekanisme CRC

2.8.1.1 Pengirim

Sebuah data yang akan dikirimkan akan dibagi terlebih dengan pembagi. Pembagi mempunyai n+1 bit. Sebelum dibagi, data yang diberi tambahan nilai 0 (nol) sebanyak n bit di belakang data. Hasil sisa dari pembagian tersebut dinamakan nilai CRC. Setelah itu nilai CRC ini akan menggantikan nilai 0 tambahan pada data tersebut. Data yang ditambahkan dengan nilai CRC itulah yang dikirimkan.

2.8.1.2 Penerima

Ketika data dengan nilai CRC itu telah sampai di penerima maka akan dibagi dengan pembagi yang sama dengan di pengirim. Jika hasil pembagian tersebut adalah 0 (nol) maka data tersebut benar. Jika tidak berarti data yang dikirimkan terdapat kesalahan pada pengiriman atau data tersebut tidak dikirim oleh pengirim yang seharusnya.



Gambar 2.17 Mekanisme CRC

2.8.2 Algoritma Dasar CRC

CRC dapat kita selesai secara matematis dengan menggunakan *Module* 2 aritmatika yaitu menggunakan penghitungan biner tanpa *carry* sehingga menggunakan operasi XOR.

Contoh *Module* 2 aritmatika:

Penjumlahan	Pengurangan
0+0=0	0-0=0
0+1=1	0-1=1
1+0=1	0-1=1
1+1=0	1-1=0

Untuk melakukan perhitungan CRC maka didefinisikan:

$$T = 2^n M + C$$

T = (k + n) bit frame untuk ditransmisi, dengan n < k

M = k bit *message*, k bit pertama dari T

C = n bit CRC, n bit terakhir dari T

P = *pattern* dari n+1 bit (pembagi)

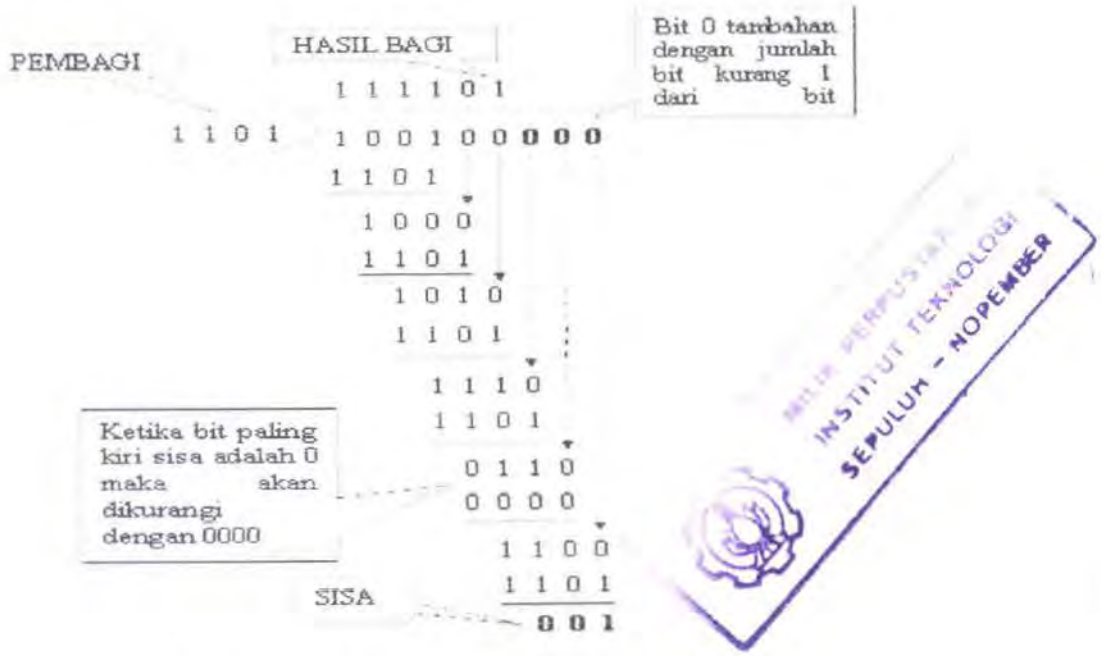
Contoh:

Diketahui: Message M = 100100

Pembagi P = 1101

Penyelesaian

1. Nilai CRC ada n bit dimana n harus bernilai kurang 1 dari bit P sehingga nilai n = 4 -1 = 3.
2. Nilai M ditambahkan nilai 0 sebanyak n bit menjadi 100100000.
3. Kemudian nilai M dibagi dengan P hingga mendapatkan nilai sisa. (Gambar 2.18)
4. Nilai M ditambahkan nilai sisa didapatkan nilai T. Sehingga nilai T menjadi 100100001. T inilah yang akan ditranmisikan menuju penerima.



Gambar 2.18 Proses Penghitungan CRC

Jika tidak ada kesalahan, maka penerima akan menerima T secara utuh. Untuk memastikan penerima menerima T yang benar, maka nilai T dibagi

Gambar 2.19 Proses Pemeriksaan CRC

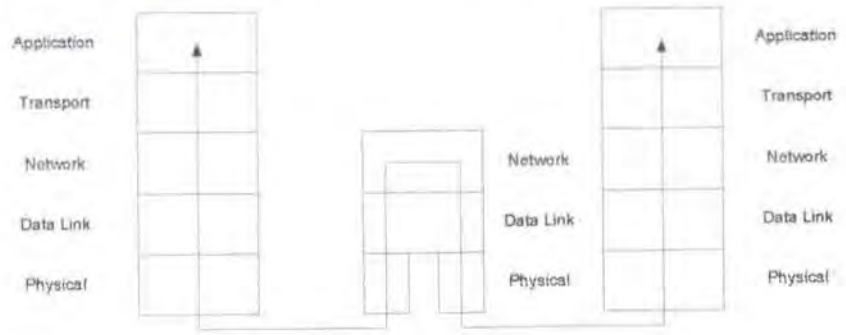
2.8.3 CRC Polynomial

Bilangan biner dapat kita ubah menjadi bentuk *polynomial*. Misalnya 10010 menjadi $1 \times ^{4} + 0 \times ^{3} + 0 \times ^{2} + 1 \times ^{1} + 0 \times ^{0} = 1 \times ^{4} + 1 \times ^{1}$. Dengan cara ini kita dapat mengubah data dan pembagi menjadi bentuk *polynomial*. Berikut standard dari pembagi dalam bentuk *polynomial*:

CRC-12
$X^{12} + X^{11} + X^{3} + X^{2} + X + 1$
CRC-6
$X^{16} + X^{15} + X^{2} + 1$
CRC-CCITT
$X^{16} + X^{12} + X^{5} + 1$
CRC-32
$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$

2.9 Router

Route yang berarti jalur adalah sebuah bagian yang tak dapat dipisahkan untuk kita berhubungan dengan network lain bahkan dengan internet. Setiap *host* didalam jaringan akan membutuhkan routing untuk berhubungan. Sebagai contoh sederhana sebagai analogi adalah bila anda akan berjalan keluar dari rumah A dengan tujuan rumah B anda harus melewati sebuah pintu keluar (routing default rumah A) kemudian berjalan misalkan melewati jalan AB (routing network A-B) hingga anda akan menemukan rumah B dan bertemu dengan si B. Demikian pula halnya dalam sebuah jaringan komputer untuk menuju Internet. Perangkat yang melakukan routing dan berfungsi sebagai jalan menuju network lainnya di sebut Router. Router bekerja pada lapisan *data link* dan lapisan *physical*.



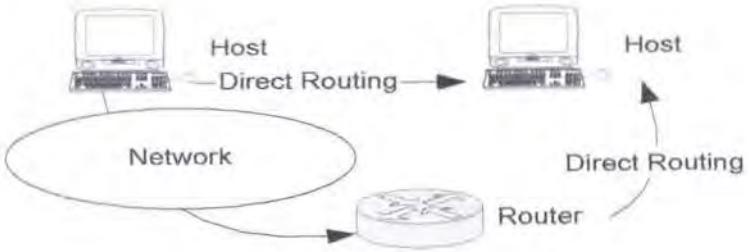
Gambar 2.20 Wilayah Kerja Router

2.9.1 Jenis Routing

Dalam implementasinya di LAN dan Internet dikenal dua jenis *routing*, yaitu:

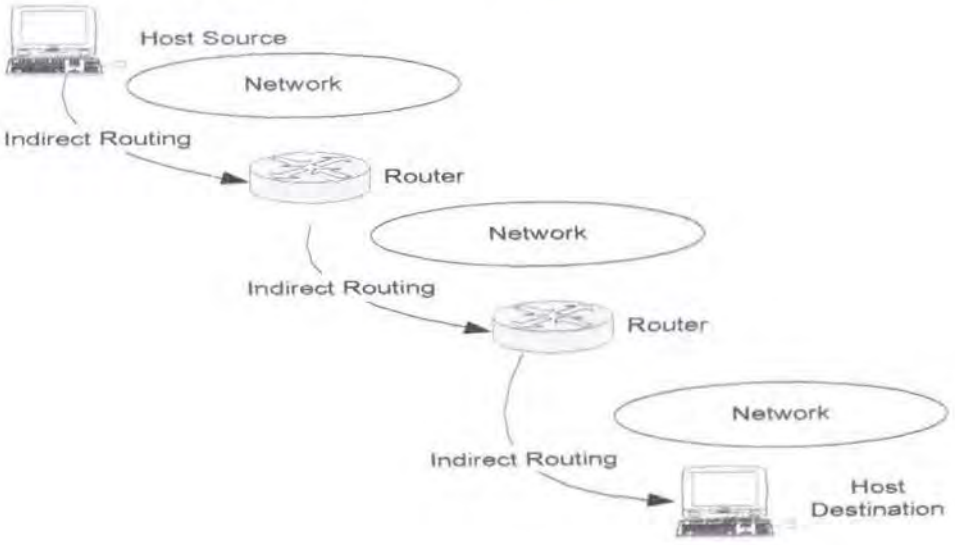
a. *Routing* Langsung (*direct routing*) dimana setiap paket akan langsung dikirimkan ketujuannya karena *remote host* biasanya masih berada

dalam network / subnetwork dengan *local host*. *Routing* ini akan terjadi bila antar *host* dalam satu subnet berkomunikasi atau *host* dengan router dalam satu subnet.



Gambar 2.21 Direct Routing

b. *Routing* Tidak Langsung (*indirect routing*) biasa ditemukan dalam network yang memiliki beberapa subnetwork atau koneksi antar network di internet. Routing ini terjadi apabila *host* pada subnet satu berhubungan dengan *host* pada subnet lain.



Gambar 2.22 Indirect Routing

2.9.2 Routing Table

Router memiliki *routing table* untuk memungkinkan komunikasi antar subnet berlangsung dan *routing default* dari *host* di masing-masing network akan diarahkan pada router yang terdapat pada network tersebut.

Routing table biasanya memiliki ketujuh bagian berikut ini : *mask*, *destination address*, *gateway address*, *flags*, *reference-count*, *use* dan antarmuka.

- **Mask.** Bagian ini mendefinisikan *mask* yang diterapkan ke alamat IP tujuan dari paket untuk menemukan alamat network atau subnetwork dari tujuan. Dalam *host*-specific dan *default routing*, mask-nya adalah 255.255.255.255. Sedangkan dalam sebuah *unsubnetted network*, *mask*-nya adalah *default mask* untuk kelas IP. Untuk kelas A adalah 255.0.0.0, kelas B adalah 255.255.0.0 dan untuk kelas C adalah 255.255.255.0.
- **Destination Address (alamat tujuan).** Bagian ini mendefinisikan alamat *host* tujuan (*host-specific address*) ataupun *network address* tujuan (*network-specific address*). Sebuah *host-specific* address tujuan memberi alamat tujuan lengkap, *net ID* dan *host*ed. Sebuah *network-specific address* hanya memberi alamat dari network dimana *entity* tujuan terhubung.
- **Gateway Address**. Bagian ini mendefinisikan alamat lokal gateway dari *host* ini atau ke subnet lain.
- **Flags**. Bagian ini mendefinisikan lima flag. Flag adalah *on/off switches* yang ada ataupun tidak. Kelima flag adalah **U** (*up*), **G** (*gateway*), **H** (*host*-specific), **D** (*added by redirection*), dan **M** (*modified by redirection*).

a. U (*up*). *Flag* U mengindikasikan router sedang *up* dan *running*. Jika *flag* ini tidak ada, berarti router ini tidak berfungsi. Paket tidak bisa diteruskan dan harus dibuang.

b. G (gateway). *Flag* G berarti bahwa tujuan berada di jaringan lain. Paket harus dikirimkan ke *next-hop router* untuk pengiriman (pengiriman tak langsung). Ketika *flag* ini hilang, berarti tujuan berada dalam jaringan ini (pengiriman langsung).

c. H (*host*-specific). *Flag* H mengindikasikan bahwa *entry* pada bagian tujuan adalah *host-specific address*. Ketika *flag* ini hilang, berarti alamat hanyalah *network address* dari tujuan.

d. D (*added by redirection*). *Flag* D mengindikasikan bahwa informasi routing untuk alamat tujuan ini sudah ditambahkan ke *routing table host* dengan *redirection message* dari ICMP.

e. M (*modified by redirection*). *Flag* M mengindikasikan bahwa informasi *routing* untuk alamat tujuan ini telah diubah dengan *redirection message* dari ICMP.

- **Reference count**. Bagian ini memberikan jumlah user yang menggunakan rute ini setiap saat. Sebagai contoh, jika lima orang pada waktu yang bersamaan terhubung ke *host* yang sama dari router ini, maka nilai untuk kolom ini adalah lima.
- **Use**. Bagian ini menunjukkan jumlah paket yang dikirimkan melalui router ini untuk tujuan yang bersesuaian.

- **Interface (antarmuka).** Bagian ini menunjukkan nama dari *interface* yang digunakan untuk *routing*.

2.10 HSRP

2.10.1 Definisi HSRP

Berdasarkan RFC2281, *Hot Standby Router Protocol* (HSRP) merupakan suatu mekanisme protokol yang bertujuan untuk memungkinkan *host* menggunakan satu router dan mempertahankan koneksi meskipun *hop* (pertemuan antara dua buah router) pertama router yang digunakan gagal. Mekanisme ini terjadi pada beberapa router yang bergabung dalam protokol ini dan membentuk satu virtual router. Protokol menjamin hanya satu router yang mengirim paket atas nama virtual router.

2.10.2 HSRP Cisco

HSRP ini pada mulanya dikenalkan oleh Cisco sebagai salah satu perusahaan pembuat router berbentuk perangkat keras. Pada bab ini akan dijelaskan HSRP dari router Cisco. Berikut ini beberapa istilah pada HSRP Cisco:

- Router aktif : router yang mengirim paket untuk virtual router
- Router standby : router yang akan menggantikan router aktif jika mengalami kegagalan.
- Grup standby : sekelompok router yang bekerja untuk menampilkan gambaran dari satu virtual router ke *host* melalui LAN.
- Hello Time : interval waktu antara pengiriman pesan *hello* dari

salah satu router.

- Hold Time : interval waktu antara penerima pesan *Hello* dan asumsi awal bahwa router pengirim gagal.

Protokol HSRP menyediakan mekanisme untuk menentukan router aktif dan router *standby*.berdasarkan nilai prioritas dari masing-masing router. Untuk melakukan mekanisme ini, semua router dalam satu kelompok tersebut saling mengirimkan pesan. Untuk mengurangi lalu lintas jaringan, hanya router aktif dan *standby* yang mengirim secara periodik pesan HSRP sekali setelah protokol menyelesaikan proses pemilihan.

HSRP menyediakan mekanisme yang dibuat untuk mendukung *non-disruptive failover* dari lalu lintas IP pada suatu keadaan. Hal yang utama adalah protokol melindungi dari kemungkinan kegagalan dari *hop* pertama router ketika komputer pengirim tidak mengenali alamat IP dari *hop* pertama router secara dinamis. Hal ini dilakukan ketika router aktif gagal, router *standby* dapat mengambil alih tanpa interupsi pada koneksi *host*, Selain itu, router *standby* juga mengambil alih pengiriman paket dari router aktif.

Pada LAN *multiple hot* grup *standby* mungkin *coexist* (hidup pada waktu yang sama) dan *overlap*. Tiap *standby* group memulai satu virtual router. Tiap grup *standby* dialokasikan satu alamat MAC, mirip seperti alamat IP. Alamat IP seharusnya menjadi *primary* subnet dalam pengunaannya pada LAN, tetapi harus berbeda dari alamat yang dialokasikan sebagai alamat *interface* untuk semua router dan *host* pada LAN, termasuk virtual alamat IP untuk grup HSRP yang lain.

Jika beberapa grup menggunakan satu LAN, pemisahan dapat dilakukan dengan cara mendistribusi *host* menjadi *standby* group yang berbeda. Pada kasus multiple group, tiap grup beroperasi *independent* terhadap yang lain. Individual router dapat berpartisipasi dalam *multiple* grup. Pada kasus ini, router mengatur state berbeda dan waktu untuk tiap grup.

2.10.3 Protokol

Dengan grup *standby*, router secara periodik memasang informasi state menggunakan berbagai macam pesan. Pesan ini termasuk dalam satu paket *Standby* protokol beroperasi pada atas UDP dan menggunakan port number 1985. Paket dikirim ke alamat *multicast* 224.0.0.2 dengan TTl 1. Router menggunakan alamat IP-nya sebagai alamat sumber untuk protokol paket, bukan alamat IP virtual. Hal ini penting sehingga HSRP router dan mengidentifikasi satu sama lain.

2.10.4 Format Paket

Format data pada UDP *datagram* sebagai berikut:

Version	Op Code	State	Hellotime
Holdtime	Priority	Group	Reserved
Authentication Data			
Authentication Data			
Virtual IP Address			

Gambar 2.23 Format Paket HSRP Cisco

Version : 1 byte

Merupakan versi pesan HSRP.

Op Code : 1 byte

Op code menyatakan tipe pesan yang ada dalam paket. Nilai *Op Code* antara lain:

0 – Hello

Pesan *Hello* dikirim untuk memberitahu bahwa router sedang bekerja dan siap menjadi router aktif ataupun router *standby*.

1 – Coup

Pesan *Coup* dikirm ketika router ingin menjadi router aktif.

2 – Resign

Pesan *Resign* dikirim ketika router tidak lagi menginginkan menjadi router aktif.

State : 1 byte

Tiap router dalam grup *standby* mengimplementasikan mesin *state*. *State* menyatakan *state* pada saat router mengirim pesan . Nilai *state* antara lain:

0 – Initial

1 – Learn

2 – Listen

4 – Speak

8 – Standby

16 – Active

Hellotime : 1 byte

Bagian ini hanya berarti pada pesan *Hello*. *Hellotime* berisi perkiraan periode antar pesan *Hello* yang dikirm router. Waktu dinyatakan dalam satuan detik. Jika *Hellotime* tidak dikonfigurasi pada router, akan diambil dari pesan *Hello* dari router aktif. *Hellotime* hanya diambil dari *hello* router aktif, jika tidak ada *Hellotime* yang dikonfigurasi dan pesan *Hello* adalah asli. Router yang mengirim pesan *Hello* harus mengisi *Hellotime*. Jika *Hellotime* tidak diambil dari

pesan *Hello* dari router aktif dan *Hellotime* tidak secara *manual* dikonfigurasi. Sebaiknya nilai *default* 3 (tiga) detik.

Holdtime : 1 byte

Bagian ini hanya berarti pada pesan *Hello*. Berisi waktu dimana pesan *Hello* seharusnya valid. Waktu dinyatakan dalam satuan detik.

Jika router mengirim pesan *Hello*, maka *receiver* harus memastikan pesan *Hello* adalah valid dalam satu *Holdtime*. *Holdtime* seharusnya paling sedikit 3 kali dari nilai *Hellotime* dan harus lebih besar dari *Hellotime*. Jika *Holdtime* tidak dikonfigurasi pada router, maka akan diambil dari pesan *Hello* dari router aktif.

Router yang dalam keadaan *state* aktif tidak boleh mengambil nilai baru untuk *Hellotime* dan *Holdtime* dari router lain, meskipun mungkin router tersebut akan melanjutkan untuk menggunakan nilai yang diambil itu dari router aktif sebelumnya. Jika *Holdtime* tidak diambil dari pesan *hello* dan tidak secara manual dikonfigurasi, nilai default yang direkomendasikan adalah 10 detik.

Priority : 1 byte

Bagian ini digunakan untuk memilih router aktif dan *standby* router. Ketika membandingkan prioritas dari du router yang berbeda, router dengan nilai prioritas tertinggi yang menang. Dalam kasus router dengan prioritas yang sama, router dengan alamat IP yang lebih tinggi yang menang.

Group : 1 byte

Bagian ini mengidentifikasikan grup *standby*. Untuk *Token Ring*, nilai antara 0 dan 2 adalah valid. Untuk media yang lain nilai antara 0 dan 255 adalah valid.

Authentication Data : 8 byte

Bagian ini berisi *clear-text* karakter *reused password*. Jika tidak ada autensifikasi data yang dikonfigurasi, direkomendasikan nilai *default* 0x63 0x69 0x73 0x63 0x6F 0x00 0x00 0x00.

Virtual IP Address : 4 byte

Alamat IP virtual yang digunakan oleh grup. Jika Alamat IP virtual tidak dikonfigurasi pada router, maka akan diambil dari pesan *Hello* dari router aktif.

2.10.5 Parameter HSRP

Informasi ini harus diketahui oleh tiap router yang termasuk grup *standby*.

- Nomor grup *standby*
- Alamat MAC virtual
- Prioritas
- Authentication Data
- *Hellotime*
- *Holdtime*

Informasi ini harus diketahui paling sedikit satu router di masing-masing grup *standby* dan bisa diketahui oleh router lain yang ada dalam grup.

Jika sebuah router memiliki prioritas lebih tinggi daripada router aktif dan *preemption* dikonfigurasi, router tersebut akan mengambil alih sebagai router aktif menggunakan pesan *Coup*.

2.10.6 State

Tiap router dalam group berpartisipasi di protokol dengan

mengimplementasikan *state machine* sederhana. Spesifikasi ini menggambarkan perilaku yang *visible* secara eksternal dari *state machine* ini. Semua router mulai dengan *initial state*. Berikut keterangan masing-masing *state*:

1. Initial

   Meruapakan *state* awal dan menandakan HSRP sedang tidak berjalan. *State* ini terjadi ketika perubahan konfigurasi atau ketika *interface* pertama kali keluar.

2. Learn

   Router tidak menentukan alamat IP virtual, dan belum mengetahui pesan *Hello* yang autentik dari router aktif. Pada *state* ini, router tetap menunggu untuk mendengar dari router aktif

3. Listen

   Router mengetahui alamat IP virtual namun tidak mengetahui siapa router aktif dan *standby* sehingga router *listen* pesan *Hello* dari router lain.

4. Speak

   Router mengirim secara periodik pesan *Hello* dan secara aktif berpartisipasi dalam pemilihan router aktif dan/atau *standby*. Router tidak dapat mengisi *state Speak* kecuali memiliki alamat IP virtual.

5. Standby

   Router merupakan kandidat untuk menjadi router aktif selanjutnya dan mengirim secara periodik pesan *Hello*. Di luar kondisi *transient*, maka harus ada maksimal satu router dalam grup yang berada dalam *state standby*.

6. Active

Router yang mengirim paket yang dikirim ke alamat MAC group. Router mengirim pesan *Hello* secara periodik. Di luar kondisi *transient*, maka harus ada maksimal satu router dalam group yang berada dalam *state active*.

2.10.7 Timer

Tiap router mengatur tiga *timer*, yaitu *timer* aktif, *timer standby*, dan *timer Hello*. *Timer* aktif digunakan untuk memonitor router aktif. *Timer* aktif dimulai pada saat pesan *Hello* autentikasi dikenali oleh router aktif. Akan dihapus pada saat *Holdtime* dikenali.

Timer standby digunakan untuk memonitor router *standby*. *Timer standby* dimulai saat pesan *Hello* autentik dikenali oleh router *standby*. Akan dihapus pada saat *Holdtime* dikenali.

Timer Hello selesai, satu kali tiap periode *Hellotime*. Jika router dalam keadaan *speak* , *standby* dan *active*, router harus membuat pesan *Hello* selama *timer Hello* habis.

2.10.8 Event

Berikut event pada mesin state HSRP:

a. HSRP dikonfigurasi pada *enable interface*.
b. HSRP di-*disable* pada *interface* atau *interface* di-*disable*.
c. *Timer* aktif habis. *Timer* aktif diset pada *Holdtime* ketika pesan *Hello* terakhir dikenali dari router aktif.
d. *Timer standby* habis. *Timer standby* diset pada *Holdtime* ketika pesan *Hello* terakhir dikenali dari router *standby*.

e. *Timer Hello* habis. *Timer* periodik untuk pengiriman pesan *Hello* yang telah habis.

f. Menerima pesan *Hello* dengan prioritas lebih tinggi daripada router dalam *state speak*.

g. Menerima pesan *Hello* dengan prioritas lebih tinggi daripada router aktif.

h. Menerima pesan *Hello* dengan prioritas lebih rendah daripada router aktif.

i. Menerima pesan *Resign* dari router aktif.

j. Menerima pesan *Coup* dari router dengan prioritas tertinggi.

k. Menerima pesan *Hello* dari router *standby* dengan prioritas tertinggi.

l. Menerima pesan *Hello* dari router *standby* dengan prioritas terendah.

2.10.9 Action

Bagian ini menjelaskan aksi yang harus dilakukan sebagai bagian dari mesin state.

A. Start Active Timer

Jika aksi ini muncul, sebagai hasil penerima pesan *Hello* autentik dari router aktif, *timer* aktif di set pada bagian *Holdtime* pada pesan *Hello*. Sebaliknya *timer* aktif di set menjadi nilai *Holdtime* yang digunakan router ini. *Timer* aktif dimulai.

B. Start Standby Timer

Jika aksi ini terjadi sebagai hasil penerima pesan *Hello* autentik dari router *standby*, *timer standby* di set pada bagian *Holdtime* pada pesan *Hello*. Sebaliknya *timer standby* di set menjadi nilai *Holdtime* yang digunakan router ini. *Timer standby* dimulai.

C. Stop Active Timer

Timer aktif dimatikan.

D. Stop Standby Timer

Timer standby dimatikan.

E. Mempelajari parameter

Aksi ini dilakukan ketika pesan autentik diterima dari router aktif. Jika alamat IP virtual untuk grup ini tidak secara manual dikonfigurasi, alamat IP virtual akan diambil dari pesan. Router akan mempelajari *Hellotime* dan *Holdtime* pesan.

F. Mengirim pesan *Hello*

Router mengirim pesan *Hello* dengan *state*nya, *Hellotime* dan *Holdtime*.

G. Mengirim pesan Coup

Router mengirim pesan *Coup* untuk menginformasikan kepada router aktif bahwa terdapat router dengan prioritas lebih tinggi.

H. Mengirim pesan Resign

Router mengirim pesan *Resign* untuk memungkinkan router lain menjadi router aktif.

I. Mengirim pesan Gratuitous ARP

J. Router melakukan broadcast paket respon ARP *advertising* alamat IP virtual dan alamat MAC virtual grup. Paket dikirim menggunakan alamat MAC virtual sebagai sumber alamat MAC pada *header layer link* bersama dengan paket ARP.

2.10.10 Transisi State

Tabel berikut menggambarkan perubahan mesin state. Untuk tiap kejadian dan state router, router harus melakukan serangkaian aksi yang dispesifikasikan dan perubahan state. Jika aksi tidak dispesifikasikan, tidak aksi yang perlu dilakukan. Jika tidak ada perubahan state yang dispesifikasikan, tidak perubahan state yang harus dilakukan.

Notasi yang digunakan pada table berikut adalah aksi yang dilakukan dilambangkan huruf. State selanjutnya dengan lambang nomor. Tanda slash ('/') membatasi aksi dan state. Contoh notasi AB/2|3 berarti pada state tersebut harus melakukan aksi A dan B kemudian berpindah ke state 2 (learn) atau 3(listen).

Tabel 2. 3 Transisi State Cisco

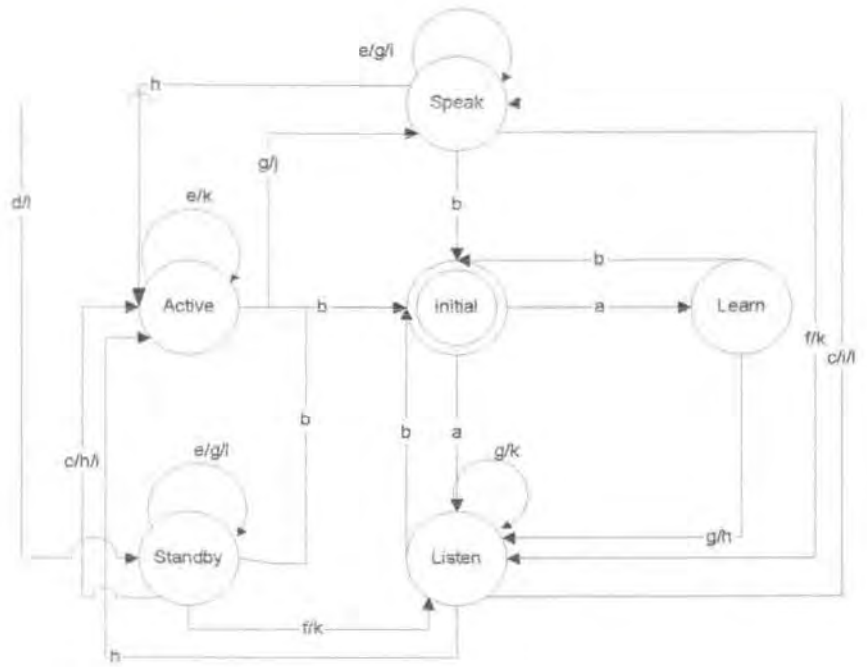
	1 Initial	2 Learn	3 Listen	4 Speak	5 *Standby*	6 Active
Event						
a	AB/2|3+					
b		CD/1	CD/1	CD/1	CD/1	CDH/1
c			AB/4		CDFI/6	
d			B/4	D/5		
e				F	F	F
f				B/3	B/3	
g		EAB/3	EA	EA	EA	AB/4
h		EAB/3	A|BGFI/6*	A|BGFI/6*	A|BGFI/6*	G
i			AB/4	A	CFI/6	
j						ABH/4
k			B	B/3	B/3	B
l			B/4	D/5		B

Keterangan:

+ Jika alamat IP virtual dikonfigurasi, maka akan diberi nilai *state* 3 (*Listen*). Namun jika alamat IP virtual tidak dikonfigurasi, bernilai *state* 2 (*Learn*). Pada kasus lain lakukan aksi A dan B.

* Jika router dikonfigurasi sebelum lakukan aksi B, G, H, dan I dan nilai *state* 6 (Active). Jika router tidak dikonfigurasi untuk sebelumnya, lakukan aksi A dengan tidak ada perubahan *state*.

Tabel transisi diatas tergambar state machine (Gambar 3.2) dimana notasi slash ('/') memisahkan antar *event*.



Gambar 2.24 State Transisi HSRP Cisco

2.10.11 Pemilihan Alamat MAC

Tiap grup HSRP memiliki alamat MAC virtual. Pada jaringan *token ring*, alamat ini menjadi alamat fungsional. Tiga alamat 0xC0 0x00 0x00 0x01 0x00

0x00, 0xC0 0x00 0x00 0x02 0x00 0x00, 0xC0 0x00 0x00 0x04 0x00 0x00 menyatakan grup 0, 1, dan 2.

Pada media yang lain, alamat MAC virtual adalah 0x00 0x00 0x0C 0x07 0xAC XX dimana XX menyatakan nomor grup HSRP. Router yang mengimplementasikan HSRP harus menggunakan alamat MAC HSRP sebagai alamat grup MAC virtual.

Router aktif harus menerima dan mengirim lalu lintas yang menjadi tujuan alamat MAC virtual grup. Router harus berhenti menerima atau mengirim lalu lintas tersebut ketika router meninggalkan state *active*.

Jika dan hanya jika router dalam keadaan aktif, router harus menggunakan alamat grup MAC virtual sebagai alamat sumber untuk pesan *Hello*. Hal ini perlu untuk memungkinkan bridge *learning* dapat mengetahui segmen LAN yang mana yang memiliki alamat MAC virtual tersebut.

Untuk tiap grup, memiliki satu alamat IP virtual dan satu alamat MAC virtual. Situasi ini muncul, karena table ARP diisi pada stasiun akhir tidak memerlukan perubahan waktu seperti pada router aktif HSRP berubah dari satu router ke yang lain.

HSRP yang bekerja pada lingkungan menggunakan bridge, bridge harus dapat segera melakukan *update* dirinya seperti perpindahan alamat IP virtual. Meskipun *learning bridge* dapat melakukannya, beberapa bridge memiliki masalah dalam hal ini. Oleh karena itu, disarankan agar *true learning bridge* digunakan dengan HSRP.

Pergerakan alamat MAC virtual dapat menyebabkan efek yang tidak

diinginkan pada lingkungan dimana *state* tambahan sangat ketat terhadap alamat MAC. Contoh pada *Token Ring*, jika *Source Route Bridging* digunakan, RIF akan disimpan dengan alamat MAC virtual ke *host* RIF cache. RIF mengindikasikan jalur dan ring akhir yang digunakan untuk mencapai alamat MAC. Transisi router menjadi *state active*, tidak akan mengubah RIF *cache* pada *host* di bridged ring. Hal ini menyebabkan paket bridged di *ring* untuk router aktif sebelumnya.

Dalam situasi tersebut, router dapat menggunakan alamat MAC normal sebagai alamat MAC virtual. Pada metode ini, alamat IP virtual akan di map ke alamat MAC yang lain. Hal ini dapat menimbulkan masalah pada stasiun akhir karena tabel ARP secara statik melakukan *mapping* antara alamat MAC dan alamat IP. Tabel ARP ini secara normal diperbarui ketika stasiun akhir menerima respon ARP tak beralasan yang dihasilkan oleh router yang mengisi *state active*.

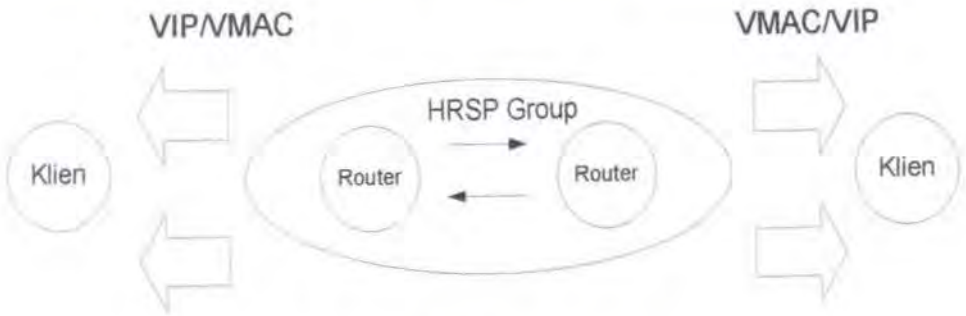
BAB III

DESAIN DAN IMPLEMENTASI SISTEM

3.1 Definisi Sistem

Aplikasi ini berjalan pada sekelompok router yang terdapat dalam satu subnet. Pada router-router tersebut dijalankan aplikasi yang sama. Tidak ada perbedaan aplikasi antara router aktif dan *standby*. Aplikasi sendiri yang akan mengatur mekanisme yang terjadi antara router aktif dan standby.

Mekanisme tersebut dapat terjadi karena router-router tersebut berhubungan secara *broadcast* pada port tertentu. Sehingga router *standby* dapat mengetahui jika sewaktu-waktu router aktif mati atau tidak dapat lagi menjalankan fungsinya. Router-router tersebut masing-masing memiliki alamat IP yang berbeda. Namun ketika menjadi aktif, router tersebut harus menggunakan alamat IP virtual yang telah ditetapkan karena alamat IP virtual ini digunakan oleh klien-klien sebagai alamat IP router mereka. Selain alamat IP virtual , router aktif akan menggunakan alamat MAC virtual. Sehingga klien tidak perlu mengetahui mekanisme yang terjadi kelompok router tersebut.



Gambar 3.1 Sistem HSRP

Untuk mengatasi kesalahan dalam pengiriman data, sistem ini dilengkapi dengan *Cyclic Redundancy Checking* (CRC). Dimana data yang akan dikirim diberi tambahan nilai CRC sehingga ketika data telah diterima akan diperiksa dengan generator CRC.

Pada sistem ini tidak meimplementasikan HSRP seluruhnya karena keterbatasan dari sebuah router komputer, yaitu router aktif tidak dapat mengirimkan pesan resign yang bertujuan memberi tahu router lain agar segera menggantikannya. Oleh karena itu sistem ini dinamakan **semi hot standby router protocol.**

3.2 Desain Sistem

3.2.1 Desain Aplikasi

3.2.1.1 Transisi State

Tiap router dalam grup yang berpartisipasi pada protokol ini dapat digambarkan dalam *state machine* sederhana. Berikut ini *state-state* yang dilalui oleh router:

1. Initial

   Merupakan state awal dan mekanisme belum berjalan. Pada saat ini router melakukan konfigurasi protokol HSRP-nya sendiri.

2. Listen

   Router belum mengetahui apakah ada router lain yang telah aktif atau tidak. State ini terjadi pada saat aplikasi baru dijalankan.

3. Speak

Kondisi ini terjadi ketika router aktif mati. Setelah itu semua router standby melakukan pemilihan router aktif.

4. Standby

   Pada saat ini router dapat menggantikan router aktif jika router aktif gagal dan mengirim pesan 'S' secara periodik pada router aktif.

5. Active

   Pada saat ini router dalam kondisi aktif dan mengirim pesan 'A' jika ada router standby yang mengirimkan pesan 'S' atau 'R'.

State pada aplikasi ini hampir sama dengan HSRP Cisco, hanya *state learn* yang ditiadakan. *State learn* tidak berfungsi karena aplikasi ini akan berjalan jika semua parameter telah dikonfigurasi dan jika setelah aplikasi berjalan telah ada router lain yang aktif, maka nilai konfigurasi virtual IP dan virtual MAC diambil dari pesan router aktif.

Untuk menuju *state* satu ke *state* lain ada *event* yang terjadi pada router tersebut. Berikut ini *event-event* yang terjadi pada router:

a. HSRP dikonfigurasi.
b. Waktu registrasi habis dan tidak menerima pesan dari router lain.
c. Waktu registrasi habis dan menerima pesan 'A'.
d. Waktu registrasi habis dan menerima pesan 'R' dengan prioritas lebih tinggi.
e. Waktu registrasi habis dan menerima pesan 'R' dengan prioritas tertinggi dan sama, namun menerima nilai IP lebih tinggi.

f. Waktu registrasi habis dan menerima pesan 'R' dengan prioritas tertinggi, sama dan nilai IP tertinggi.

g. Waktu standby habis dan tidak menerima pesan 'A'.

h. Waktu standby habis dan menerima pesan 'A'.

i. Menerima pesan 'S'.

j. Program error atau dihentikan.

Ketika *event* berlangsung diperlukan *action* sebelum menuju ke state berikutnya. Action pada aplikasi ini tidak melakukan pengiriman pesan *Coup* dan *Resign* seperti HSRP Cisco. Hal tesebut dilakukan karena router yang sedang dalam kondisi aktif tidak perlu diubah menjadi standby yang akan berakibat pada ketidakstabilan lalu lintas jaringan. *Action* pada router yang menggunakan aplikasi ini adalah sebagai berikut:

A. Memulai waktu registrasi

Action ini muncul pada awal aplikasi setelah *state initial* dan ketika router aktif mati. *Action* dilakukan setelah mengirimkan pesan 'R'.

B. Memulai waktu standby

Action ini dilakukan oleh router *standby* yang akan menunggu pesan dari router aktif.

C. Mengirim pesan 'R'

Action ini muncul pada awal aplikasi setelah *state initial* untuk mengetahui apakah ada router aktif dan pemilihan antar router *standby*.

D. Mengirim pesan 'A'

Action ini dilakukan oleh router aktif yang menerima pesan 'S dari router *standby*.

E. Mengirim pesan 'S'

Action ini dilakukan oleh router *standby* untuk mengetahui apakah router aktif masih hidup.

F. Ubah nilai VIP dan VMAC sama dengan VIP dan VMAC router aktif.

Action ini dilakukan ketika router mendapat pesan 'A' dari router aktif.

G. Mengubah nilai MAC menjadi VMAC dan menambahkan nilai VIP

Action ini dilakukan oleh router yang akan menjadi router aktif.

H. Mengubah nilai VMAC menjadi MAC dan membuang nilai VIP

Action ini dilakukan oleh router yang akan menjadi router *standby*. Selain itu aplikasi ini dilakukan jika program mengalami *error* atau terhentikan.

I. Mengembalikan *routing table*

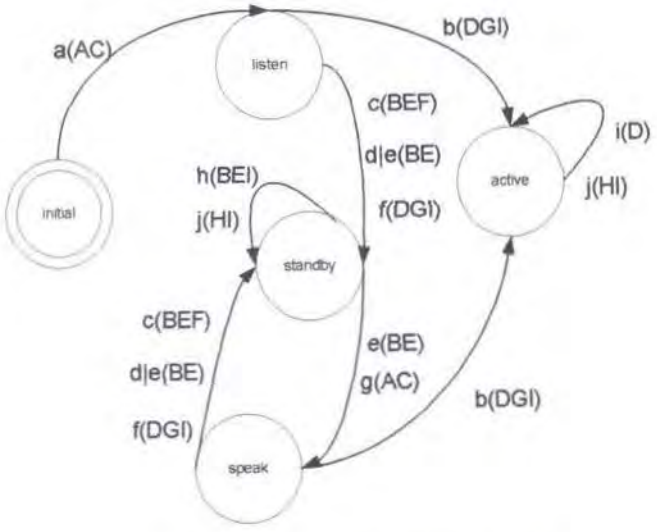
Action ini dilakukan oleh router yang mengalami perubahan status dari *standby* ke aktif atau sebaliknya.

Hubungan antara *state*, *event* dan *action* dapat digambarkan pada tabel 3.1. Untuk setiap *event* dan *state* router harus melakukan serangkaian *action*. Notasi angka menunjukkan *state* berikutnya, huruf *lowercase* menunjukkan *event* dan huruf *uppercase* menunjukkan *action*. Untuk mengetahui keterangan masing-masing notasi (angka dan huruf) lihat keterangan di atas. Tanda *slash* (/) memisahkan antara *action* dan *state* berikutnya.

Tabel 3. 1Transisi State

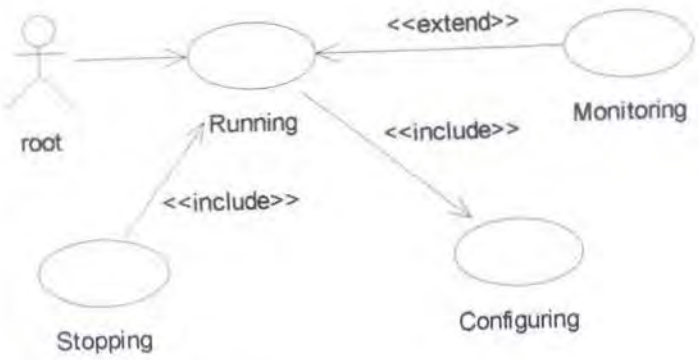
	1 Initial	2 Listen	3 Speak	4 Standby	5 Active
Event					
A	AC/2				
B		DGI/5	DGI/5		
C		BEF/4	BEF/4		
D		BE/4	BE/4		
E		BE/4	BE/4	BE/3	
F		DGI/4	DGI/4		
G				AC/3	
H				BEI/4	
I					D
J				HI	HI

Untuk memperjelas table 3.1 tersebut, dapat digambarkan *state machine* pada Gambar 3.2. Yang terdapat dalam tanda kurung merupakan aksi yang harus dilakukan untuk menuju ke state berikutnya,



Gambar 3.2 State Machine Insco

3.2.1.2 Use Case Diagram



Gambar 3.3 Use Case Diagram Aplikasi

Use Case Diagram diatas menggambarkan aplikasi ini memiliki sebuah aktor yaitu *root* (*administrator*). *Root* dapat melakukan proses *monitoring*, *running*, *stopping* dan *configuring*. Berikut penjelasan masing-masing *use case*:

- Monitoring

  Pada *use case* ini root dapat melihat perubahan mekanisme yang terjadi pada router tersebut seperti router menjadi aktif, router menjadi *standby*, pemilihan router dan kesalahan-kesalahan konfigurasi. *Use case* ini terjadi jika use case running telah berjalan.

- Running

  Use case ini akan berjalan jika *use case configuring* telah dilakukan. Dapat dikatakan *use case* ini merupakan *use case* utama karena saat *use case* ini terjadilah mekanisme HSRP tersebut.

- Stopping

  Root dapat menghentikan proses running pada *use case* ini. Oleh karena itu *use case* ini harus didahului *use case running*. Namun proses

running juga dapat dihentikan jika terjadi *error* selama aplikasi ini berjalan pada *background* proses.

- Configuring

  Proses konfigurasi parameter untuk *use case* running terjadi pada *use case* ini.

3.2.1.3 Activity Diagram

Berikut adalah *activity diagram* dari aplikasi ini yang didasarkan atas use case yang ada.

1. Monitoring

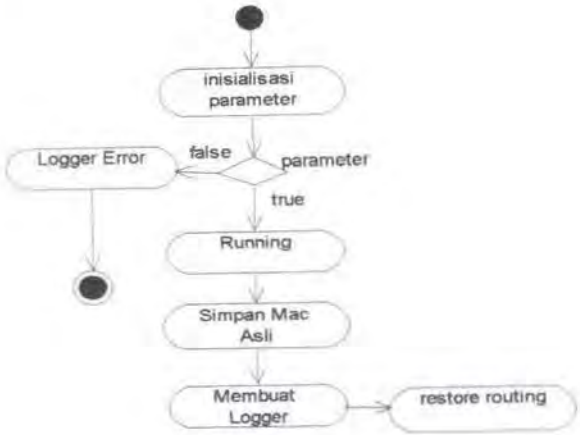
Pada *activity diagram monitoring*, *root* perlu melakukan pengecekkan apakah proses tersebut sedang *running* atau tidak hal ini bisa dilakukan dengan perintah ps –ef. Jika proses tersebut berjalan maka *root* dapat melihat *logger* aplikasi yang terdapat pada *syslog*.

Gambar 3.4 Activity Diagram Monitoring

2. Running

Pada *activity diagram* ini, parameter yang telah melalui proses *configuring* akan diinisilisasi. Jika ada kesalahan parameter yang diberikan misalnya di router

tersebut hanya ada eth0 dan eth1 tapi aplikasi ini menerima input eth3 maka proses *running* akan dihentikan. Namun jika semua parameter cocok maka proses akan segera menyimpan alamat MAC asli, membuat *logger* sebagai tanda awal mekanisme dan mengembalikan nilai routing yang benar. *Activity diagram* tedapat pada gambar 3.5.



Gambar 3.5 Activity Diagram Running

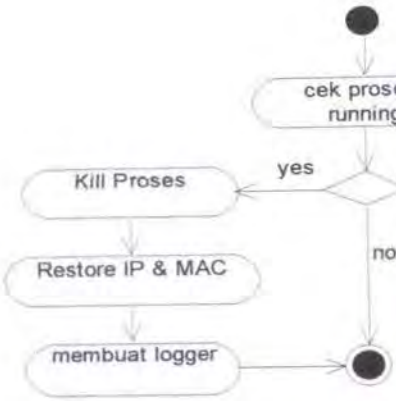
3. Stopping

Pada *use case* ini *root* mengecek apakah proses tersebut berjalan atau tidak jika sedang berjalan maka proses dapat dimatikan. Namun sebelum proses tersebut dimatikan maka ada *handler* untuk mengembalikan router ke keadaan semula baik berupa alamat IP maupun MAC. Selain *root*, aplikasi juga dapat terhenti jika terjadi *error* selama proses *background* berlangsung. Gambar use case ini dapat dilihat pada gambar 3.6.

4. Configuring

Use case ini digambarkan pada Gambar 3.7. Pada *use case* ini *root* memasukkan input-input yang dibutuhkan. Kemudian program melakukan

pengecekkan apakah input yang diberikan benar. Jika benar maka akan diterusan ke proses running. Jika salah maka program akan memberikan pesan kesalahan dan *root* harus menuliskan konfigurasi ulang. Selain memasukkan input langsung pada program. *Root* juga bisa menuliskan file konfigurasinya terlebih dahulu. Kemudian program dapat melakukan pembacaan file konfigurasi tersebut. Jika ada kesalahan pada file konfigurasi, proses *running* akan dihentikan.



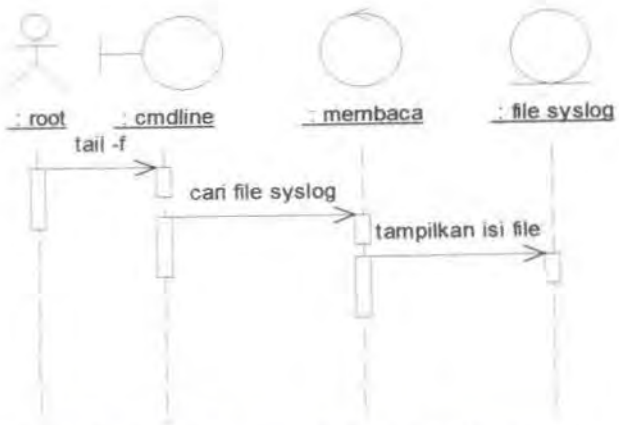
Gambar 3.6 Activity Diagram Stopping

Gambar 3.7 Activity Diagram Configuring

3.2.1.4 Sequence Diagram

1. Sequence Diagram Monitoring

Proses *monitoring* hanya dapat dilakukan pada *command line*. *Root* memberikan perintah tail –f untuk file syslog. Dengan perintah tersebut akan diketahui hal-hal yang terkait dari aplikasi tersebut, baik status router, pesan kesalahan dan lainnya. Prose monitoring ini terdapat pada Gambar 3.8.



Gambar 3.8 Sequence Diagram Monitoring

2. Sequence Diagram Running

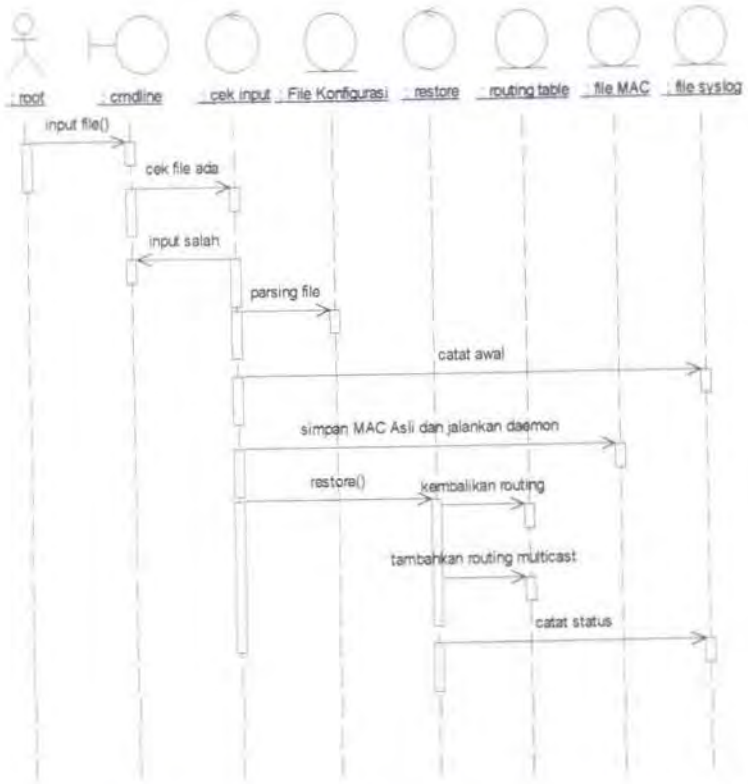
Pada *command line*, proses *running* (Gambar 3.9) dapat dilakukan dengan konfigurasi langsung atau memasukan file konfigurasi. Jika nilai input atau hasil *parsing* file konfigurasi benar, maka akan dilakukan proses pencatatan pada file syslog. Setelah itu nilai MAC asli disimpan pada sebuah file dan *daemon* dijalankan. Pengembalian nilai *routing* dan penambahan *routing multicast* pada *routing table* merupakan proses berikutnya.

Sedangkan untuk proses running yang menggunakan *form* (Gambar 3.10), root menekan tombol start dan langkah selanjutnya sama dengan *sequence diagram* yang menggunakan *command line*.

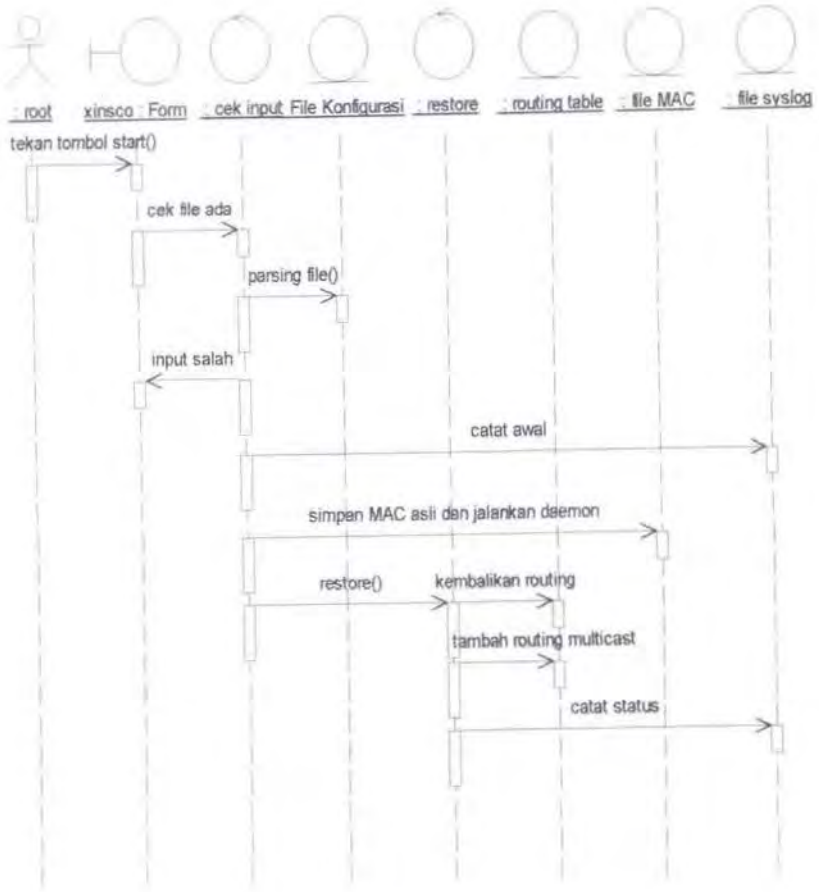
3. Sequence Diagram Configuring

Pada *sequence diagram configuring command line* (gambar 3.11), *root* dapat melihat input apa saja yang harus diberikan dengan mencetak *help*. Setelah itu root dapat memberikan input yang sesuai. Jika input yang diberikan salah, maka akan di tampilkan pesan kesalahan. Jika benar, maka proses running akan dilakukan.

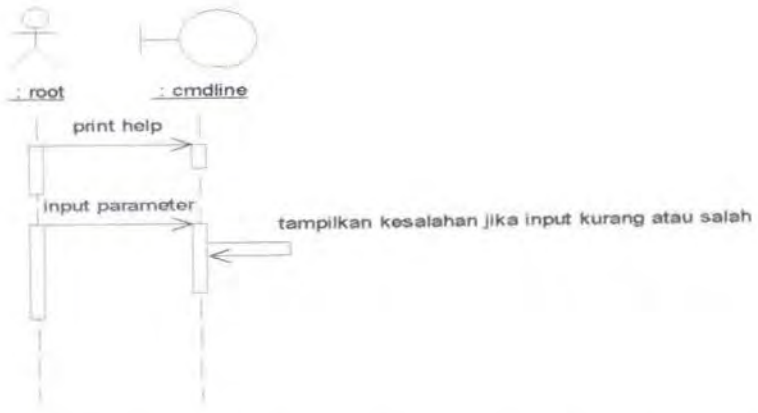
Untuk *sequence diagram configuring form* (gambar 3.12), user diberikan kemudahan dalam memasukkan input. Input dapat berupa parameter yang tersedia pada *textbox* atau memasukkan input file. Input yang diberikan oleh user akan disimpan dalam bentuk file yang akan menjadi input bagi proses running.



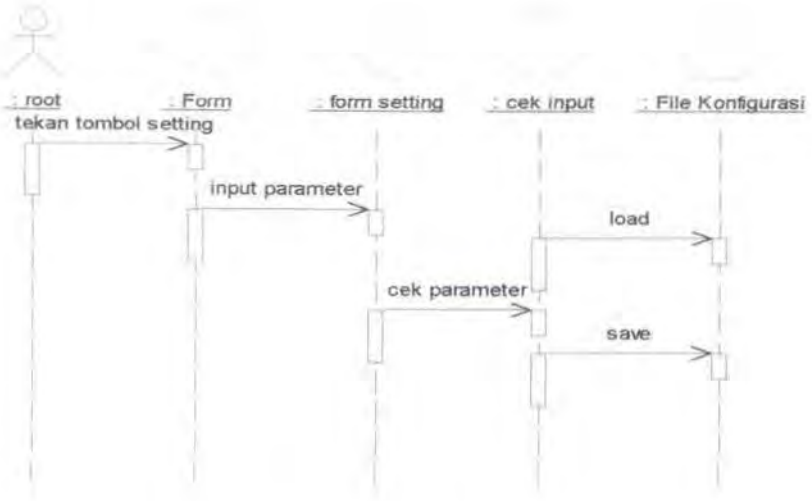
Gambar 3.9 Sequence Diagram Running Command Line



Gambar 3.10 Sequence Diagram Running Form



Gambar 3.11 Sequence Diagram Configuring Command Line

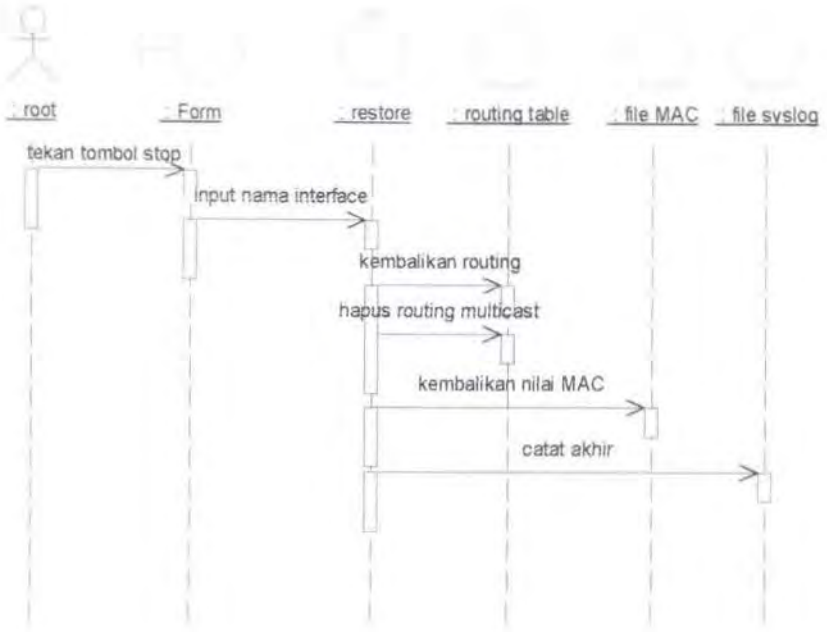


Gambar 3.12 Sequence Diagram Configuring Form

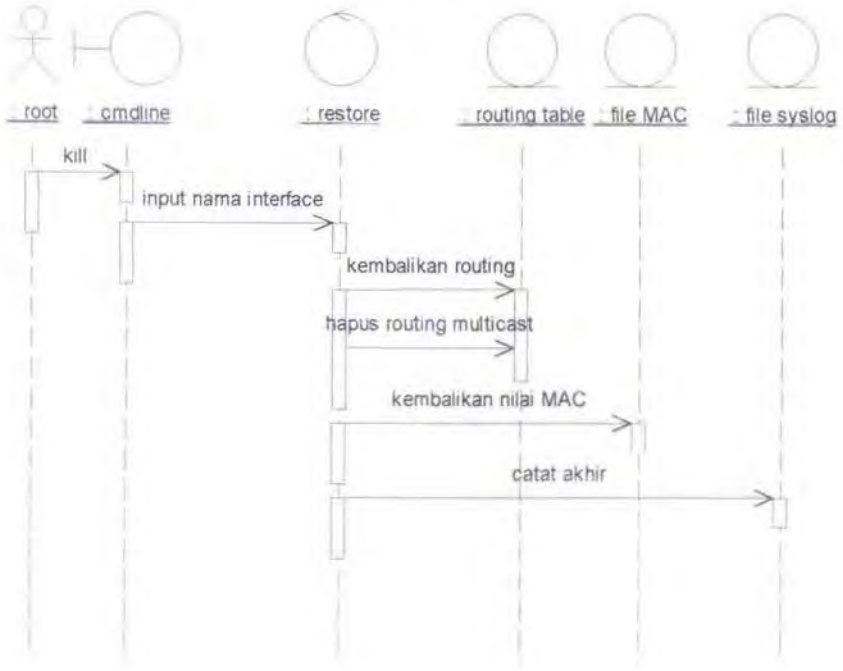
4. Sequence Diagram Stopping

Root dapat melakukan proses ini melalui *command line* atau *form*. Pada *command line* yang terdapat pada gambar 3.13, *root* mengetikkan perintah *kill*. Sedangkan pada *form* yang terdapat pada gambar 3.14, *root* menekan tombol stop. Selain *root*, aplikasi ini juga dapat melakukan penghentian sendiri jika terjadi *error* pada saat proses *running*. Misalnya tiba-tiba *interface down*.

Ketika proses *stopping* dilakukan, secara otomatis dikembalikan alamat IP dan MAC kembali seperti semula sebelum program ini dijalankan. Hal ini dilakukan agar tidak terjadi konflik dengan router standby yang akan menggantikan router ini dengan nilai MAC dan IP yang sama. Selain itu, dilakukan juga pencatatan pada *syslog* bahwa router ini telah berhenti pada jam, menit dan detik saat itu.



Gambar 3.13 Sequence Diagram Stopping



Gambar 3.14 Sequence Diagram Stopping

3.3 Implementasi Sistem

3.3.1 Kebutuhan Sistem

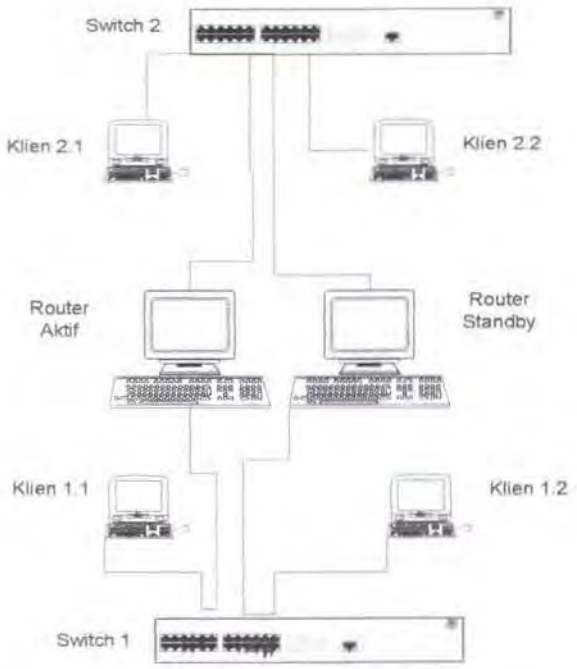
3.3.1.1 Router

Aplikasi router ini tersedia dalam dua macam yaitu *Command Line Interface* dan *Graphical User Interface*. User dapat memilihnya berdasarkan hardware yang dimilikinya. Kebutuhan hardware untuk router yang menggunakan aplikasi ini secara umum adalah memiliki 2 LAN CARD dengan sistem terinstal OS Linux dengan kernel minimum 2.2 yang mendukung multicast. Berikut kebutuhan masing-masing aplikasi :

1. Aplikasi *Command Line Interface* membutuhkan hardware dengan spesifikasi minimun PC Pentium 133, RAM 32 MB, Hardisk 1GB dengan tersedia gcc minimal versi 3.2. Sedangkan untuk file rpm dan deb tidak membutuhkan gcc dan harus ada adalah *library* C yang mendukung *socket*.
2. Aplikasi *Graphical User Interface* membutuhkan hardware dengan spesifikasi minimun PC Pentium II, RAM 64 MB, Hardisk 2,3GB dengan aplikasi *Command Line Interface* (nomor 1) telah terinstal. Aplikasi ini juga membutuhkan *library* GTK dan XWindows yang terinstal.

3.3.1.2 Klien

Kebutuhan untuk klien adalah dengan spesifikasi hardware minimun PC Pentium 386, RAM 32 MB, Hardisk 100 MB dan terhubung dengan LAN yang satu subnet dengan salah satu router. Hubungan router dan klien terdapat pada Gambar 3.15.



Gambar 3.15 Kebutuhan Server dan Klien

3.4 Implementasi Aplikasi

Aplikasi ini yang menerapkan HSRP ini diberi nama INSCO. Berikut hal-hal yang berkaitan dengan aplikasi INSCO.

3.4.1 Instalasi INSCO

Aplikasi INSCO dapat diinstal pada semua distro Linux. Aplikasi ini tersedia dalam 3 macam paket, yaitu tar.gz, rpm dan deb. Paket tar.gz dapat diinstal pada semua distro yang terdapat gcc, paket rpm khusus dinstal pada distro yang mendukung Red Hat *Packet Manager* seperti Red Hat, Mandrake, Fedora dan lain-lain, sedangkan paket deb khusus diinstal pada distro yang mendukung paket debian seperti debian dan knoppix. Berikut ini cara penginstalan paket-paket tersebut.

3.4.1.1 Tar.gz

1. Ekstrak file insco-1.0.tar.gz atau xinsco-1.0.tar.gz (untuk *Graphical User Interface*) pada path /usr/local/src

```
# cd /usr/local/src
# tar xvfz insco-1.0.tar.gz
```

2. Lakukan konfigurasi penginstalan

```
# ./configure -options=options value
Untuk mengetahui pilihan parameter untuk konfigurasi.
# ./configure --help
```

3. Lakukan kompilasi pada paket.

```
# make
Lakukan penginstalan paket
# make install
```

3.4.1.2 RPM

Untuk paket rpm, instalasi cukup mudah namun tidak fleksibel seperti tar.gz.

```
# rpm -ivh insco-1.0.rpm
```

3.4.1.3 DEB

Seperti paket rpm, instalasi paket debian juga mudah.

```
# dpkg -i insco_1.0.deb atau # dpkg -i xinsco_1.0.deb
```

Jika menggunakan APT, dengan catatan debian anda terhubung dengan source debian di internet.

```
# apt-get -install insco_1.0 atau  apt-get -install xinsco_1.0
```

3.4.2 Format Data Protokol HSRP INSCO

Dalam berhubungan router dalam kelompok HSRP menggunakan protocol INSCO. Struktur protocol INSCO disusun berdasarkan format seperti gambar 3.16.

msg	IP				VIP				VMAC						Prior itas	CRC	
R/A/S	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xxx	xx	xx	xx	xx	xx	xx	Xxx	xxx	xxx
1 byte	← 4 byte →				← 4 byte →				← 6 byte →						1byte	←2byte→	

Gambar 3.16 Kerangka Format Protocol INSCO

Keterangan :

1. **msg**

   Msg bertipe data char. Msg dapat bernilai 'R','S' dan 'A'. Msg bernilai 'R' berarti registrasi router

2. **IP**

   IP bertipe data string yang terdiri atas 4 karakter. Setiap karakter bernilai antara 0 sampai dengan 255. IP ini bernilai sesuai dengan alamat IP router pengirim.

3. **VIP**

   VIP bertipe data string yang terdiri atas 4 karakter. Setiap karakter bernilai antara 0 sampai dengan 255. VIP ini bernilai sesuai dengan alamat IP virtual kelompok HSRP.

4. **VMAC**

VMAC bertipe data string yang terdiri atas 6 karakter. Setiap 2 karakter yang berdampingan bernilai antara 0 sampai dengan 255. VMAC ini bernilai sesuai dengan virtual MAC Address kelompok HSRP.

5. **prioritas**

   Prioritas bertipe data char. Prioritas bernilai antara 0 sampai dengan 255. Nilai menunjukkan tingkat prioritas sebuah router dibandingkan dengan router lainnya pada saat terjadi pemilihan router. Semakin tinggi prioritas maka semakin besar peluang sebuah router untuk mengantikan router aktif jika mengalami kegagalan.

6. **CRC**

   CRC bertipe data string yang terdiri atas 2 karakter. CRC ini berfungsi sebagai pemeriksa kesalahan data yang diterima. Penghitungan CRC ini didapatkan dari seluruh data paket yang dikirimkan sebelum ditambhakan CRC.

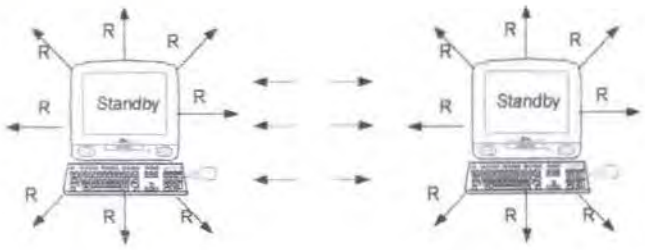
3.4.3 Mekanisme Sistem INSCO

3.4.3.1 Kondisi Sistem

Dalam mekanisme sistem INSCO terdapat 3 kondisi yang dapat terjadi yaitu :

1. Semua router dalam kondisi *standby*. Hal ini dapat terjadi karena router baru dinyalakan bersama-sama atau router aktif mati sehingga terjadi pemilihan di antara router-router yang ada. Semua router dalam kondisi ini akan mengirimkan msg bernilai 'R' hal ini dilakukan untuk mengetahui

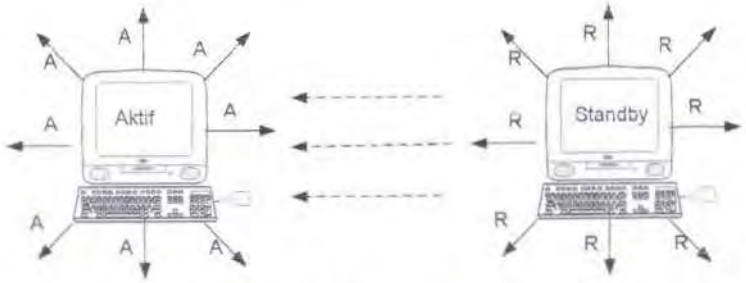
nilai prioritas dan nilai alamat IP terakhir yang menjadi pertimbangan untuk menentukan siapa yang akan menjadi router aktif selanjutnya. Nilai prioritas lebih diutamakan daripada nilai alamat IP terakhir. Jika ada router memiliki nilai prioritas tertinggi daripada router lain maka router ini akan menjadi aktif. Tetapi jika ada router lain yang memiliki prioritas yang sama maka akan dipilih berdasarkan nilai alamat IP terakhir. Misalnya ada router A memiliki prioritas 1 dan alamat IP 10.126.10.169, sedangkan router B memiliki prioritas 1 dan alamat IP 10.126.10.168, maka router A akan dipilih menjadi router aktif. Gambar 3.18 memgambarkan bagaimana proses pemilihan router.



Gambar 3.17 Kondisi Pemilihan Router

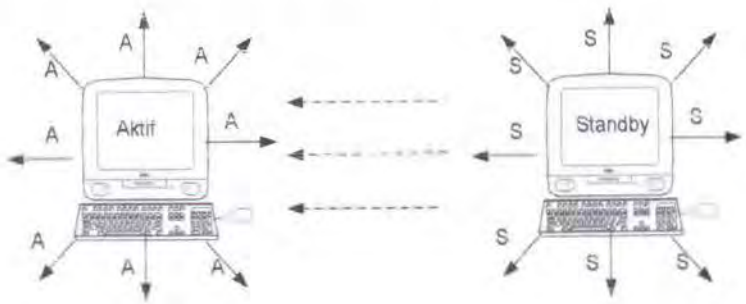
Gambar 3.18 Proses Pemilihan Router

2. Sebuah router baru dihidupkan kemudian router ini mengirimkan msg bernilai 'R'. Sebuah router aktif yang menerima pesan tersebut segera memberikan balasan msg bernilai 'A' sebagai pemberitahuan bahwa sudah ada router yang aktif. Jika router tersebut tidak menerima msg bernilai 'A' maka router ini akan menjadi aktif.



Gambar 3.19 Router Standby

3. Router aktif dan *standby* sudah ada. Setiap selang beberapa detik router standby mengirimkan msg bernilai 'S'. Sebuah router aktif yang menerima pesan tersebut segera memberikan balasan msg bernilai 'A' sebagai pemberitahuan bahwa dirinya masih hidup.



Gambar 3.20 Router aktif dan standby

3.4.4 Pseudo Code Utama

3.4.4.1 Pseudo Code Mengubah Alamat IP

```
int soket
struct ifreq ipbaru
struct sockaddr_in *ip_addr
soket ← socket(AF_INET, SOCK_DGRAM, 0)
 if soket<0 then
   print kesalahan
ipbaru.ifr_name ← ifname
ip_addr ← &ipbaru.ifr_addr
ip_addr->sin_family ← AF_INET
ip_addr->sin_addr ← vip
 if ioctl(soket, SIOCSIFADDR, &ipbaru)<0 then
   print kesalahan
```

Gambar 3.21 Pseudo Code Mengubah Alamat IP

Untuk mengubah alamat IP tidak perlu men-*Down*-kan *interface*. Hanya memasukkan nilai IP alamat baru dan SIOCIFADDR pada fungsi IOCTL Pengubahan alamat IP digunakan untuk membuat *virtual* IP.

3.4.4.2 Pseudo Code Mengubah Alamat Netmask

```
int soket
struct ifreq ipbaru
struct sockaddr_in *mask_addr
soket ← socket(AF_INET, SOCK_DGRAM, 0)
 if soket<0 then
   print kesalahan
ipbaru.ifr_name ← ifname
mask_addr ← &ipbaru.ifr_addr;
mask_addr->sin_family ← AF_INET;
mask_addr->sin_addr ← "255.255.255.0"
 if ioctl(soket, SIOCSIFNETMASK, &ipbaru)<0 then
   print kesalahan
```

Gambar 3.22 Pseudo Code Mengubah Alamat Netmask

Untuk mengubah alamat *netmask* perlu memasukkan nilai alamat *netmask* baru dan SIOCIFNETMASK pada fungsi IOCTL. Pengubahan alamat netmask

digunakan untuk membuat *virtual* IP yang membutuhkan nilai netmask yang benar.

3.4.4.3 PseudoCode Mengubah Alamat Broadcast

```
int soket
struct ifreq ipbaru
struct sockaddr_in *broad_addr
soket ← socket(AF_INET, SOCK_DGRAM, 0)
 if soket<0 then
   print kesalahan
ipbaru.ifr_name ← ifname
broad_addr ← &ipbaru.ifr_addr
broad_addr->sin_family ← AF_INET
&broad_addr->sin_addr ← bip
 if ioctl(soket, SIOCSIFBRDADDR , &ipbaru)<0 then
   print kesalahan
```

Gambar 3.23 PseudoCode Mengubah Alamat Broadcast

Untuk mengubah alamat *broadcast* perlu memasukkan nilai alamat *broadcast* baru dan SIOCIFBRDADDR pada fungsi IOCTL. Pengubahan alamat broadcast digunakan untuk membuat *virtual* IP yang membutuhkan nilai netmask yang benar.

3.4.4.4 Pseudo Code Up Interface

Pseudo code dibawah ini (Gambar 3.24) berfungsi untuk meng-*Up*-kan interface. Dimana nilai IFF_UP 0x1 merupakan nilai interface yang *Up*. Untuk meng-*Up*-kan interface pada fungsi IOCTL dimasukkab nilai SIOCSIFFLAGS.

Pada aplikasi ini interface yang sudah diubah alamat MAC-nya harus di-*up*-kan agar dapat berfungsi kembali sebagai jalur komunikasi dalam jaringan.

```
int soket
struct ifreq *macup
soket ← socket(AF_INET, SOCK_DGRAM, 0)
 if soket<0 then
  print kesalahan
macup->ifr_name ← nameth
 if ioctl(soket, SIOCGIFFLAGS, macup) < 0 then
  print kesalahan
 else
    if macup->ifr_flags and (~IFF_UP) then
       if ioctl(soket, SIOCGIFFLAGS, macup) < 0 then
        print kesalahan
       else
          macup->ifr_flags ← macup->ifr_flags  | IFF_UP
          if ioctl(soket, SIOCSIFFLAGS, macup) < 0 then
           print kesalahan
    else
     print interface sudah up
```

Gambar 3.24 Pseudo Code Up Interface

3.4.4.5 Pseudo Code Down Interface

Pseudo code gambar 3.25 berfungsi untuk meng-*Down*-kan *interface*. Dimana nilai *complement* IFF_UP 0x1 merupakan nilai interface yang *Down*. Untuk meng-*Down*-kan Ethernet pada fungsi IOCTL dimasukkan nilai SIOCSIFFLAGS.

Proses down interface dilakukan pada saat akan mengubah alamat MAC sebenarnya ke alamat virtual MAC atau sebaliknya. Selain itu *down interface* juga dilakukan pada *interface* alamat *virtual* IP. Ketika sebuah router aktif mengalami kegagalan maka interface yang menggunakan alamat *virtual* IP harus di-*down*-kan supaya tidak terjadi konflik dengan router aktif pengganti.

3.4.4.6 Pseudo Code Mengubah Alamat MAC

Code pada gambar 3.26 bertujuan untuk mengubah nilai alamat MAC yang merupakan bawaan dari pabrik. Sebelum mengubah nilai MAC, *interface*

harus di-*Down*-kan terlebih dahulu. Kemudian mendapatkan semua nilai yang berkaitan dengan *interface* dengan memasukkan nilai SIOCGFHWADDR ke fungsi IOCTL. Setelah itu memasukkan nilai MAC baru dan nilai SIOCSFHWADDR ke fungsi IOCTL.

Pengubahan alamat MAC pada aplikasi ini digunakan untuk membuat alamat *virtual* MAC yang menjadi alamat MAC untuk router aktif, sedangkan router *standby* tetap menggunakan alamat MAC sebenarnya

```
int soket
struct ifreq *macdown
soket ← socket(AF_INET, SOCK_DGRAM, 0);
  if soket<0 then
    print kesalahan
macdown->ifr_name ← nameth
  if ioctl(soket, SIOCGIFFLAGS, macdown) < 0 then
    print kesalahan
  else
     if macdown->ifr_flags & IFF_UP then
        if ioctl(soket, SIOCGIFFLAGS, macdown) < 0 then
           print kesalahan
        else
           macdown->ifr_flags ← macdown->ifr_flags & ~IFF_UP;
           if ioctl(soket, SIOCSIFFLAGS, macdown) <0 then
           print kesalahan
     else
      print interface sudah down
```

Gambar 3.25 Pseudo Code Down Interface

```
struct ifreq *mac_baru
int soket
soket ← socket(AF_INET, SOCK_DGRAM, 0)
  if soket<0 then
    print kesalahan
mac_baru->ifr_name ← ifname
  if downeth(ifname)>=0 then
     if ioctl(soket, SIOCGIFHWADDR, mac_baru) < 0 then
       print kesalahan
     for i ← 0 to 6
        mac_baru->ifr_hwaddr.sa_data[i] ← mac[i]
        i ← i + 1
     if ioctl(soket, SIOCSIFHWADDR, mac_baru) < 0 then
       print kesalahan
  else
    upeth(ifname)
▷ up interface
```

Gambar 3.26 Pseudo Code Mengubah Alamat MAC

3.4.4.7 Pseudo Code Mengubah Routing

```
struct rtentry default_gw
struct sockaddr_in *gw ← default_gw.rt_gateway
struct sockaddr_in *dst ← default_gw.rt_dst
struct sockaddr_in *mask ← default_gw.rt_genmask
dst->sin_family ← AF_INET;
gw->sin_family ← AF_INET;
mask->sin_family ← AF_INET;
default_gw.rt_flags ← iflags;
default_gw.rt_metric ← metric;
gw->sin_addr.s_addr ← gate_addr;
dst->sin_addr.s_addr ← net_addr;
mask->sin_addr.s_addr ← mask_addr;
soket ← socket(AF_INET, SOCK_STREAM, 0)
  if soket<0 then
    print kesalahan
default_gw.rt_dev ← namaeth
  if ioctl(soket, SIOCADDRT , &default_gw)<0 then
    print kesalahan
```

Gambar 3.27 Pseudo Code Mengubah Routing

Sebagaimana biasanya dalam mengubah routing, diperlukan variable *destination*, *gateway* dan *netmask*. Untuk mengatur *routing* dibutuhkan *struct rtentry* untuk menangani data-data yang berkaitan dengan *routing table*. Jika semua data pada *struct* ini telah terisi, fungsi IOCTL dipanggil dengan nilai SIOCADDRT jika ingin menambahkan routing dan SIOCDELRT untuk menghapus *routing*.

Penambahan dan penghapusan pada *routing table* dalam aplikasi ini dibutuhkan untuk mengembalikan nilai *routing* pada *interface* yang *down* kemudian di *–up*-kan.

3.4.4.8 Pseudo Code Penghitungan Waktu

```
sigemptyset(&sigsetalrm)
sigaddset(&sigsetalrm,SIGALRM)
signal(SIGALRM, cekstatus)
▷ reset alarm
alarm(0)
 if sendto(sock, &pesanhsrp, sizeof(pesanhsrp), 0, (struct sockaddr
 *)&broadcastAddr, sizeof(broadcastAddr)) < 0 then
  print kesalahan
 else
    alarm(3)
     while(true)
        ▷ awal alarm dinyalakan
        sigprocmask(SIG_UNBLOCK,&sigsetalrm,NULL)
         if recvfrom(sock,&pesanlain,sizeof(pesanlain),0,NULL,0)< 0
         then
          print kesalahan
         else
          ▷ menerima pesan
        ▷ melakukan proses lain
        sigprocmask(SIG_BLOCK,&sigsetalrm,NULL);
        ▷ akhir alarm
```

Gambar 3.28 Code Perhitungan Waktu

Untuk melakukan perhitungan waktu memerlukan signal SIGALRM. SIGALRM ini bersifat sebagai interupsi jika telah tiba waktu yang ditentukan. Alarm dimulai setelah proses pengiriman data dilakukan dan terbatas pada proses penerimaan data dilakukan. Ketika alarm menginterupsi akan dijalankan fungsi *handling*.

3.4.5 Desain Antar Muka

Implementasi antar muka untuk aplikasi ini terbagi menjadi dua, yaitu *Command Line Interface* dan *Graphical User Interface*. Berikut ini adalah keterangan masing-masing dari desain antar muka aplikasi:

3.4.5.1 Command Line Interface

Aplikasi ini dikembangkan dengan pustaka ***gengetopt*** yang berfungsi untuk melakukan *parsing argument* dan opsi program. Untuk melakukan parsing, aplikasi ini digunakan *gengetopt* versi 2.6. Berikut keterangan dari masing-masing opsi aplikasi ini :

1. Menampilkan seluruh opsi aplikasi.

```
# insco --help atau # insco -h
```

2. Menampilkan versi dari aplikasi.

```
# insco -V atau # insco --version
```

3. Memberikan nilai pada opsi yang mebutuhkan input.

Tedapat enam buah opsi pada aplikasi ini yaitu:

- **p** atau **Port**, opsi ini harus diisi dengan nomor port yang digunakan untuk komunikasi secara *broadcast*. Sebuah router dapat menggunakan menggunakan alamat IP multicast yang sama namun nomor port harus berbeda jika router tersebut menjalankan aplikasi sebanyak jumlah *interface*-nya.
- **r** atau **Prioritas**, opsi harus diisi dengan nomor prioritas antara 1 sampai dengan 255.
- **I** atau **VIP**, opsi ini harus diisi dengan alamat *virtual* IP yang akan digunakan oleh kelompok.
- **M** atau **MAC**, opsi ini harus diisi dengan alamat *virtual* MAC yang akan digunakan oleh kelompok.
- **E** atau **ETH**, opsi ini harus disi dengan nama interface ethernet yang

digunakan, misalnya eth0, eth1 dan lain-lain.

- **T** atau **VETH,** opsi ini diisi dengan nama *virtual interface* yang digunakan untuk alamat *virtual* IP, misalnya eth0:0, eth1:1 dan lain-lain. Nama *virtual interface* untuk nilai sebelum ':' (titik dua) harus sama dengan nilai pada opsi E atau VETH.
- **F** atau **namafile**, opsi ini tidak harus diisi jika opsi-opsi diatas telah diisi dan jika opsi ini telah diisi maka opsi diatas tidak diperlukan. Opsi ini diisi dengan nama file konfigurasi.

Contoh penggunaan opsi:

```
# insco \
> -p 2000 -r 3
> -I 10.126.10.167
> -M 00:51:00:10:00:28
> -E eth0 -T eth0:0
# insco \
> -Prioritas=3 -Port=2000
> -VIP=10.126.10.167
> -MAC=00:51:00:10:00:28
> -ETH=eth0 -VETH=eth0:0
```

Gambar 3.29 menggambarkan *command line interface* setelah perintah Insco diketikkan.

```
[iin@localhost gab]$ insco -h
insco 1.0

Purpose:
  Membuat Semi Hot Standby Router Protocol

Usage: insco [OPTIONS]...
   -h         --help              Print help and exit
   -V         --version           Print version and exit
   -FSTRING   --namafile=STRING   Nama File konfigurasi
(Optional) (default='/etc/insco.conf')
   -pINT      --Port=INT          Alamat Port Untuk Broadcast
   -rINT      --Prioritas=INT     Prioritas Router
   -ISTRING   --VIP=STRING        Virtual IP Address Router
   -MSTRING   --VMAC=STRING       Virtual MAC
   -ESTRING   --ETH=STRING        Nama Ethernet contoh: eth0,
eth1
   -TSTRING   --VETH=STRING       Nama Ethernet Virtual contoh:
eth0:0, eth1:1
```

Gambar 3.29 Command Line Interface Insco

3.4.5.2 Graphical User Interface

Aplikasi yang menggunakan *Graphical User Interface* (GUI) ini dikembangkan dengan GTK+. Penggunaan aplikasi GUI ini bertujuan untuk memudahkan user dalam memasukkan input. Pada aplikasi ini terdapat dua buah *form* penting, yaitu *form* utama (Gambar 3.30) dan *setting* (Gambar 3.31).

1. Form Utama

Pada *Form* utama ini terdapat tiga buah *button* di atas, satu buah tabel dan dua buah *button* di bawah. Enam buah *button* di atas terdiri atas *button add, edit, del, help* dan *about*. Pada bagian tengah form ini terdapat tabel yang menampilkan isi dari aplikasi insco yang telah berjalan. Tabel ini terdiri atas enam buah *field*, yaitu *Interface*, *Virtual Interface*, *Virtual* IP, *Virtual* MAC, Prioritas dan Port. Sedangkan dua buah *button* di bawah terdiri atas *button start* dan *stop*.Berikut ini keterangan masing-masing dari *button* pada form utama:

1. *Button Start*

   Jika *button* ini ditekan maka akan memanggil aplikasi insco yang *command line interface*,

2. *Button Stop*

   Aksi dari *button* ini adalah menghentikan proses aplikasi insco yang sedang berjalan.

3. *Button Add*

   Aksi dari *button* ini adalah menambah isi dari tabel. Sehingga user mempunyai banyak pilihan setting Insco mana yang akan di-*running*.

4. *Button Del*

   Aksi dari *button* ini adalah menghapus isi dari tabel sesuai dengan pilihan yang dipilih pada tabel.

5. *Button Edit*

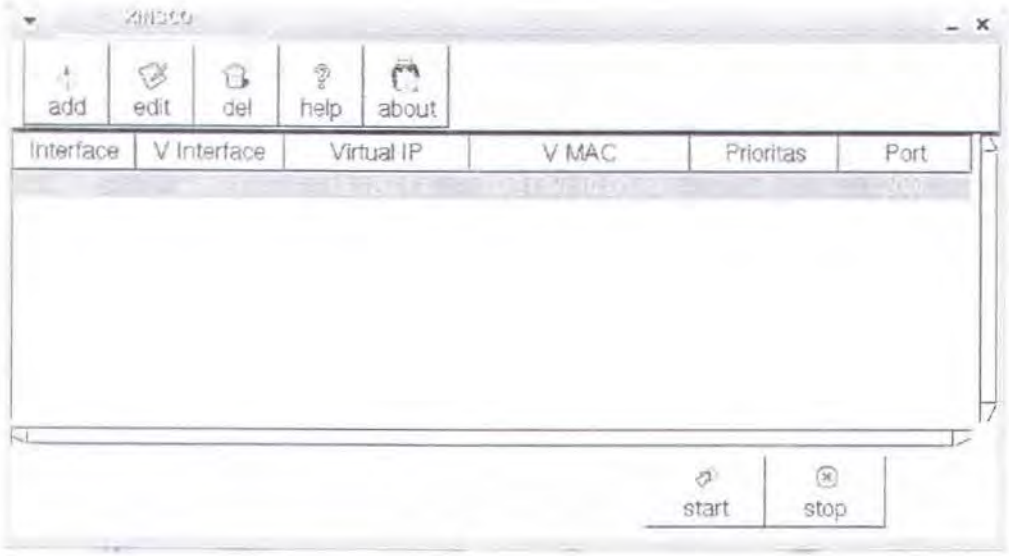
   Aksi dari *button* ini adalah menampilkan form setting sesuai dengan pilihan pada tabel.

6. *Button Help*

   Jika button ini ditekan, maka akan tampil dialog cara penggunaan aplikasi ini.

7. *Button About*

   Aksi dari *button* ini adalah menampilkan dialog about.



Gambar 3.30 Graphical User Interface Form Utama Insco

2. Form Setting

Pada form setting terdapat enam buah text box dan empat buah button. Berikut ini keterangan masing-masing text box dan button :

1. *Combo box Interface*

   User harus mengisi dengan nama Ethernet yang akan dipakai. Misalnya eth0, eth1 dan lain-lain.

2. *Text box Virtual Interface*

   Text box ini diisi dengan nama virtual Ethernet yang akan dipakai dengan ketentuan nama depan sebelum tanda ':' (titik dua) harus sama dengan *Combo box Device*.

3. *Text box Virtual IP*

   User harus menberikan nilai alamat virtual IP.

4. *Text box Virtual MAC*

   User harus menberikan nilai alamat virtual MAC.

5. *Text box* Prioritas

   Text box ini diisi nilai prioritas router dengan nilai antara 1 sampai dengan 255.

6. *Text box Port*

   Text box ini diisi nilai port yang digunakan untuk komunikasi secara *multicast*. Port ini bernilai antara 1024 sampai dengan 65535.

7. *Button Open*

   Aksi dari *button* ini adalah menampilkan *file dialog* yang digunakan untuk memilih file konfigurasi khusus untuk sebuah setting interface.

8. *Button Save*

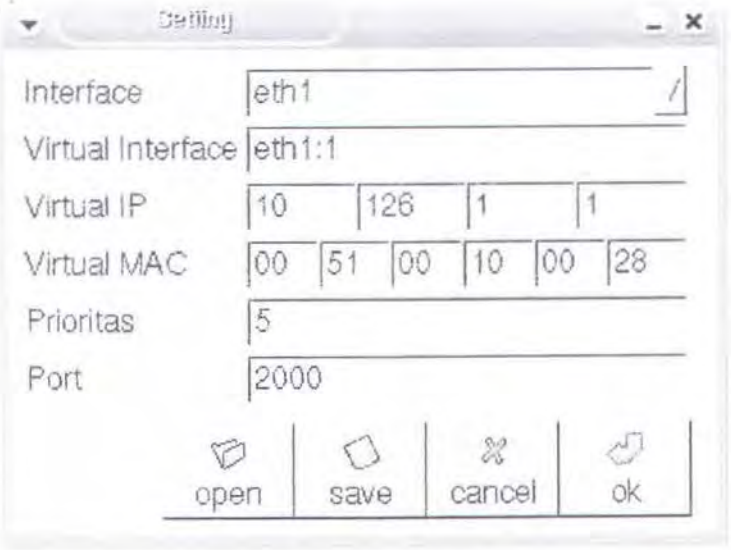
   Aksi dari *button* ini adalah menampilkan *file dialog* untuk memilih nama file yang digunakan untuk menyimpan apa yang terdapat pada *text box* dan *combo box* yang berada pada form setting ke dalam file. Hasil File konfigurasi ini dapat diambil lagi dengan *button open*.

9. *Button Cancel*

   Aksi dari *button* ini adalah membatalkan perintah *add* atau *edit* serta menutup *form setting*.

10. *Button Ok*

    Aksi dari *button* ini adalah menjalankan perintah *add* atau *edit* serta menutup *form setting*. Setelah itu hasil dari *add* dan *edit* akan terlihat pada tabel.



Gambar 3.31 Graphical User Interface Form Setting Insco

BAB IV UJI

COBA DAN EVALUASI

4.1 Lingkungan Uji Coba

Uji coba ini dilakukan pada jaringan sesungguhnya dan jaringan dengan *user mode linux* (UML).

4.1.1 Jaringan Sesungguhnya

Untuk melakukan uji coba pada jaringan sesungguhnya dibutuhkan 3 (tiga) buah komputer sebagai router, minimal 2 (dua) buah komputer sebagai klien dan 2 (dua) buah hub. Berikut spesifikasi masing-masing komputer dan hub:

4.1.1.1 Router

1. Darqueenice
   - Sistem Operasi : Linux Debian 3 Kernel 2.4.23
   - Prosesor : P-IV 2 GHz
   - RAM : 256 MB DDRAM
   - HD : 40 GB
   - LAN Card : 2 buah D-LINK
   - Alamat IP : Eth1 10.126.10.169
     Eth2 10.126.100.6
2. Falcon
   - Sistem Operasi : Linux Red Hat 9 Kernel 2.4.20
   - Prosesor : P-III GHz

- RAM : 256 MB SDRAM
- HD : 20 GB
- LAN Card : 2 buah D-LINK
- Alamat IP : Eth0 10.126.10.168
  Eth1 10.126.100.5

3. Hsrp
   - Sistem Operasi : Linux Debian 3 Kernel 2.4.23
   - Prosesor : P-IV 2 GHz
   - RAM : 1 GB DDRAM
   - HD : 40 GB
   - LAN Card : 2 buah 3COM
   - Alamat IP : Eth0 10.126.10.167
     Eth1 10.126.100.4

4.1.1.2 Klien

1. Hagemaru
   - Sistem Operasi : Windows XP
   - Prosesor : P-IV 2 GHz
   - RAM : 128 MB SDRAM
   - HD : 40 GB
   - Alamat IP : Eth0 10.126.10.205
2. Goldenboy
   - Sistem Operasi : Windows Advanced Server 2000

- Prosesor : P-III 1
- RAM : 128 MB SDRAM
- HD : 20 GB
- Alamat IP : Eth0 10.126.100.3

4.1.1.3 Hub

1. 3Com 3C16610
   - Data Transfer Rate : 10 Mbps dan 100 Mbps.
   - Jumlah port : 12 buah
2. D-Link DES 1016D
   - Data Transfer Rate : 10 dan 100 Mbps
   - Jumlah port : 16 buah

4.1.2 Jaringan dengan User Mode Linux

Uji coba pada jaringan dengan User Mode Linux (UML) dilakukan karena terbatasnya jumlah komputer yang ada. UML yang dipakai adalah versi 2.4.19 dengan rootfs kernel versi 2.4. Jaringan ini menggunakan netmask 255.255.255.240. Subnet pertama mempunyai *network address* 10.126.1.0 dan *broadcast address* 10.126.1.15, sedangkan subnet kedua mempunyai network address 10.126.1.16 dan *broadcast address* 10.126.1.31. Pada jaringan ini tidak menggunakan klien karena yang dihitung hanya kecepatan perpindahan router. Tabel 4.1 menjelaskan nilai alamat router pada masing-masing subnet.

Tabel 4. 1 Nama dan Alamat IP Router

Nama Router	Interface Eth0	Interface Eth1	Prioritas
Satu	10.126.1.2	10.126.1.19	5
Dua	10.126.1.3	10.126.1.20	4
Tiga	10.126.1.4	10.126.1.21	3
Empat	10.126.1.5	10.126.1.22	2
Lima	10.126.1.6	10.126.1.23	1

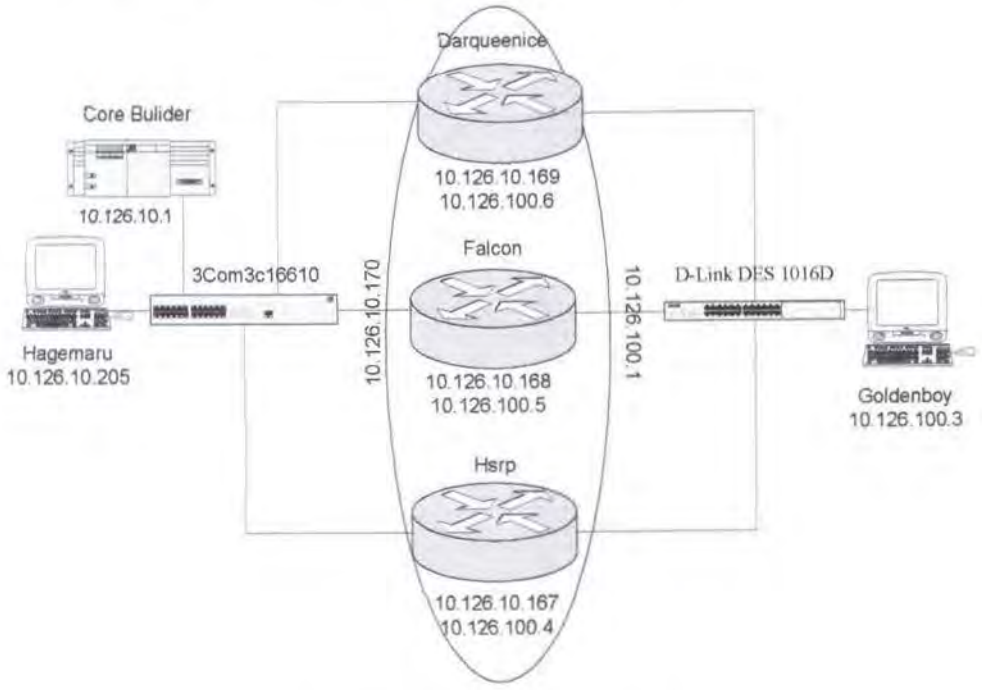
4.2 Skenario Uji Coba

Skenario uji coba dilakukan dengan menjalankan aplikasi pada semua router. Kemudian semua klien disetting routing sesuai yang diinginkan. Antara klien sama-sama melakukan proses ping atau pengiriman data, sedangkan antara router melakukan proses ping terhadap IP virtual kelompok insco. Selain itu, dijalankan juga program **Ethereal** yang akan melakukan monitor setiap paket yang masuk pada host khusus pada jaringan sesungguhnya, sedangkan pada jaringan user mode linux hanya menggunakan waktu pada syslog. Ketika klien sedang berkomunikasi, router aktif dimatikan. Masing-masing skenario pada jaringan sebenarnya dilakukan dua kali uji coba, yaitu uji coba dengan mengirimkan file dan tanpa mengirimkan file. Berikur keterangan dari skenario uji coba :

4.2.1 Uji Coba 1

Semua router mempunyai default routing gateway ke 10.126.10.1 dan semua router salah satu kabel jaringannya terhubung dengan switch jaringan network address 10.126.10.0. Dimana switch tersebut berhubungan dengan Core Builder Fakultas Teknologi Informasi. Sedangkan kabel jaringan router yang lain

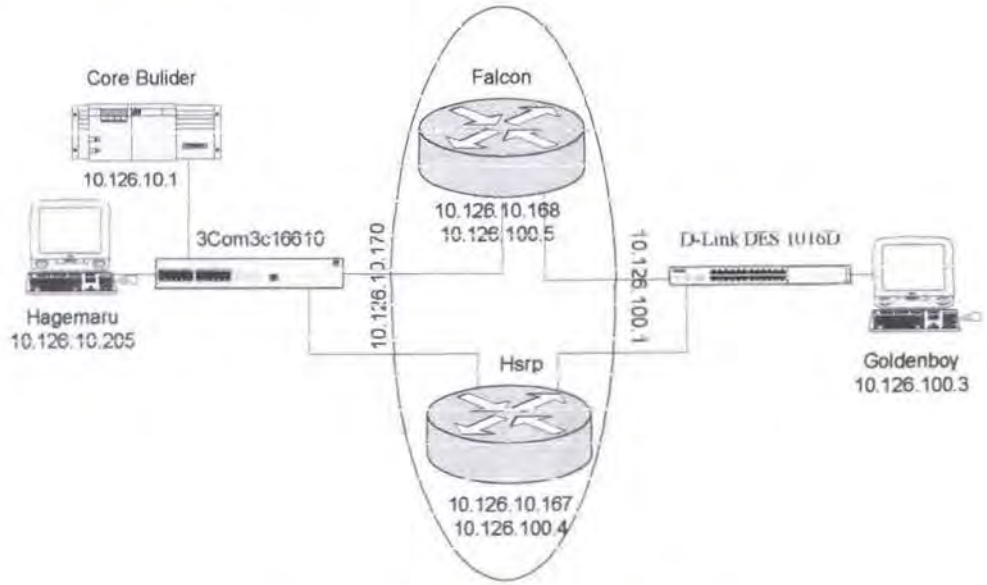
terhubung dengan switch jaringan netwok 10.126.100.0. Agar klien untuk network 10.126.100.0 dapat berhubungan dengan jaringan luar pada Core Builder ditambahkan routing untuk 10.126.100.0 melewati alamat IP 10.126.10.170. Alamat IP 10.126.10.170 ini akan menjadi alamat virtual IP kelompok router untuk network address 10.126.10.0, sedangkan alamat IP 10.126.100.1 menjadi alamat virtual IP untuk network address 10.126.100.0. Salah satu komputer menjadi klien pada network address 10.126.10.0 terhubung pada switch network address 10.126.10.0 dan diberikan nilai routing default gateway ke 10.126.10.1, sedangkan komputer klien pada network address 10.126.100.0 terhubung pada switch network address 10.126.100.0 dan diberikan nilai routing default gateway ke 10.126.100.1. Skenario ini ditunjukkan oleh gambar 4.1.



Gambar 4.1 Skenario Uji Coba 1

4.2.2 Uji Coba 2

Skenario dua sama dengan skenario pertama, namun pada skenario ini hanya dua buah router yang dijalankan, yaitu falcon dan hsrp. Skenario ini ditunjukkan oleh gambar 4.2.

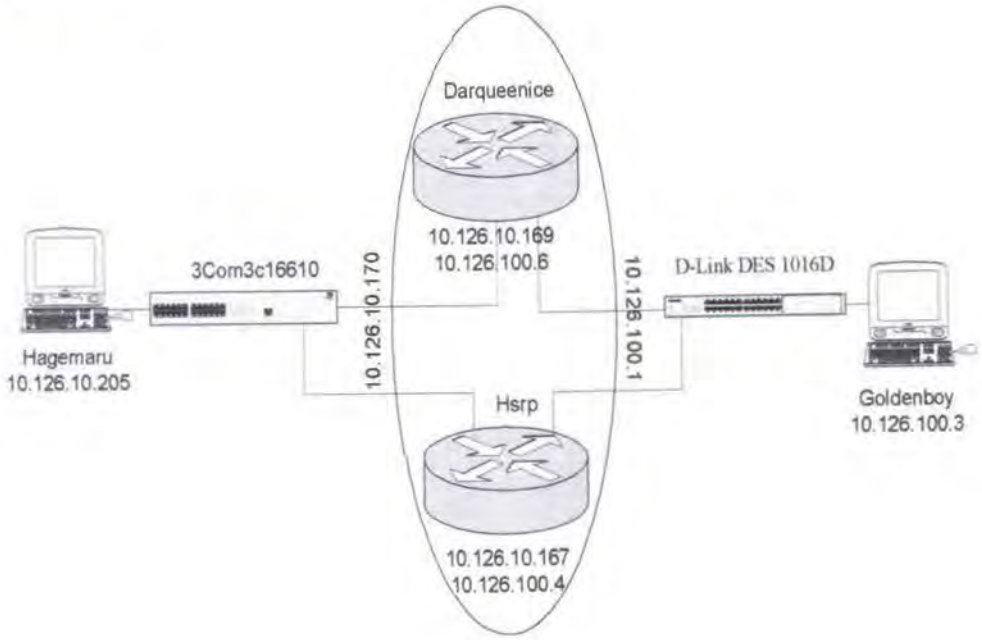


Gambar 4.2 Skenario Uji Coba 2

4.2.3 Uji Coba 3

Pada skenario ini router yang dipakai adalah darqueenice dan hsrp. Semua router tidak mempunyai routing gateway ke manapun. Semua router salah satu kabel jaringannya terhubung dengan switch jaringan network address 10.126.10.0. Sedangkan kabel jaringan router yang lain terhubung dengan switch jaringan netwok 10.126.100.0. Alamat IP 10.126.10.170 akan menjadi alamat virtual IP kelompok router untuk network address 10.126.10.0, sedangkan alamat IP 10.126.100.1 menjadi alamat virtual IP untuk network address 10.126.100.0. Salah satu komputer menjadi klien pada network address 10.126.10.0 terhubung

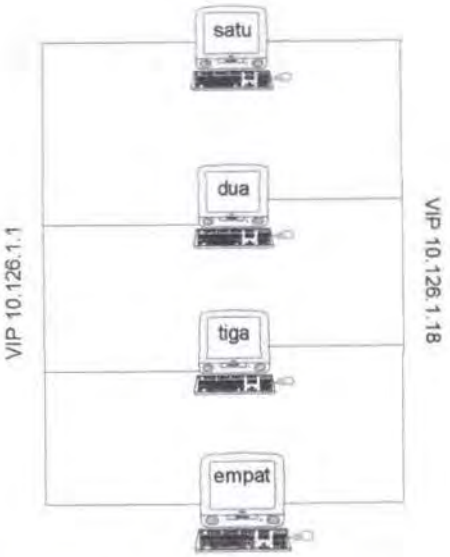
pada switch network address 10.126.10.0 dan diberikan nilai routing default gateway ke 10.126.10.170, sedangkan komputer klien pada network address 10.126.100.0 terhubung pada switch network address 10.126.100.0 dan diberikan nilai routing default gateway ke 10.126.100.1. Skenario ini ditunjukkan oleh gambar 4.3.



Gambar 4.3 Skenario Uji Coba 3

4.2.3 Uji Coba 4

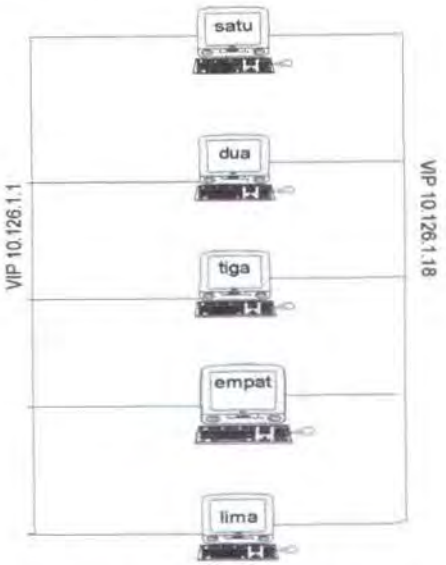
Pada skenario ini menggunakan 4 (empat) buah router dengan UML. Alamat virtual IP yang digunakan pada network pertama adalah 10.126.1.1, sedangkan pada network kedua adalah 10.126.1.18



Gambar 4. 4 Skenario Uji Coba 4

4.2.3 Uji Coba 5

Pada skenario ini menggunakan 5 (lima) buah router dengan UML. Alamat virtual IP yang digunakan pada network pertama adalah 10.126.1.1, sedangkan pada network kedua adalah 10.126.1.18



Gambar 4. 5 Skenario Uji Coba 5

4.3 Pelaksanaan Uji Coba

a. Jaringan Sesungguhnya

Ketika uji coba dijalankan program Ethereal dijalankan. Program ini berfungsi untuk menangkap setiap paket yang melewati *host* itu sendiri. Pada klien hagemaru (10.126.10.205) dilakukan Ethereal dengan filter :

```
host 10.126.100.1 or host 10.126.10.170 or host 10.126.100.3
```

Sedangkan pada setiap router dilakukan ethereal dengan filter :

```
host 10.126.100.1 or host 10.126.10.170
```

Setelah uji coba dilakukan, selanjutnya dilakukan proses penghitungan waktu *reply* dari klien ke klien, klien ke router maupun router ke router ketika terjadi perpindahan router dengan rumus :

$$\Delta T = Trp - Trq$$

ΔT = Selisih waktu paket request pertama setelah reply dengan waktu paket reply setelah request terakhir ketika terjadi perpindahan router.

Trp = Waktu paket reply setelah request terakhir ketika terjadi perpindahan router.

Trq = Waktu paket request setelah reply pertama ketika terjadi perpindahan router.

b. Jaringan UML

Ketika aplikasi dijalankan pada router dilakukan monitoring dengan perintah *tail -f /var/log/syslog*. Pada syslog akan terlihat waktu kapan aplikasi mulai jalan, pemilihan router, aplikasi mati dan lainnya. Hasil uji coba ini

berkaitan dengan selisih waktu ketika router aktif mati dengan router standby menjadi aktif.

Berikut hasil uji coba baik untuk jaringan sesungguhnya maupun jaringan UML :

4.3.1 Uji Coba 1

1. Tanpa mengirim file.

Router aktif adalah darqueenice sedangkan router standby adalah falcon dan hsrp. Klien yang digunakan adalah 10.126.10.205 dan 10.126.100.3. Gambar 4.6, 4.7 dan 4.8 merupakan hasil ethereal pada klien 10.126.10.205

No. -	Time	Source	Destination	Protocol	Info
3	2004-08-03 11:52:00.630320	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply
7	2004-08-03 11:52:01.629376	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
19	2004-08-03 11:52:07.128524	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
35	2004-08-03 11:52:12.629831	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
36	2004-08-03 11:52:12.630417	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.6 Hasil Ethereal 10.126.10.205 Ping 10.126.100.1 (1.1)

Berdasarkan gambar 4.6, router 10.126.100.1 kembali dapat terhubung setelah 11.001041 detik.

No. -	Time	Source	Destination	Protocol	Info
6	2004-08-03 11:52:01.206392	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
17	2004-08-03 11:52:06.629539	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
33	2004-08-03 11:52:12.129127	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
34	2004-08-03 11:52:12.129299	10.126.10.170	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.7 Hasil Ethereal 10.126.10.205 Ping 10.126.10.170 (1.1)

Berdasarkan gambar 4.7, router 10.126.10.170 kembali dapat terhubung setelah 10.922907 detik.

No.	Time	Source	Destination	Protocol	Info
5	2004-08-03 11:52:01.129216	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply
9	2004-08-03 11:52:02.128528	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
21	2004-08-03 11:52:07.644157	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
37	2004-08-03 11:52:13.128818	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
40	2004-08-03 11:52:13.129811	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.8 Hasil Ethereal 10.126.10.205 Ping 10.126.100.3 (1.1)

Berdasarkan gambar 4.8, klien 10.126.100.3 kembali dapat terhubung setelah 11.001283 detik.

Gambar 4.9 dan 4.10 merupakan hasil ethereal pada router hsrp.

No.	Time	Source	Destination	Protocol	Info
6	2004-08-03 11:59:51.651727	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
11	2004-08-03 11:59:52.651742	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
16	2004-08-03 11:59:53.651726	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
45	2004-08-03 11:59:58.651839	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
46	2004-08-03 11:59:58.651844	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
47	2004-08-03 11:59:58.652020	10.126.100.1	10.126.100.4	ICMP	Echo (ping) reply

Gambar 4.9 Hasil Ethereal 10.126.100.4 Ping 10.126.100.1 (1.1)

Berdasarkan gambar 4.9, Router 10.126.100.1 kembali dapat terhubung setelah 8.000142 detik.

No.	Time	Source	Destination	Protocol	Info
5	2004-08-03 11:59:50.631789	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
13	2004-08-03 11:59:51.631778	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
18	2004-08-03 11:59:52.631776	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
23	2004-08-03 11:59:53.631775	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
28	2004-08-03 11:59:54.631796	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
34	2004-08-03 11:59:55.631774	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
66	2004-08-03 12:00:00.631899	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
67	2004-08-03 12:00:00.632057	10.126.10.170	10.126.10.167	ICMP	Echo (ping) reply

Gambar 4.10 Hasil Ethereal 10.126.10.167 Ping 10.126.10.170 (1.1)

Berdasarkan gambar 4.10, Router 10.126.10.170 kembali dapat terhubung setelah 10.000268 detik.

Gambar 4.11 dan 4.12 merupakan hasil ethereal pada router falcon.

No. -	Time	Source	Destination	Protocol	Info
4	2004-08-03 12:00:10.219648	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
8	2004-08-03 12:00:11.219669	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
12	2004-08-03 12:00:12.219615	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
17	2004-08-03 12:00:13.219561	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
23	2004-08-03 12:00:14.219609	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
29	2004-08-03 12:00:15.219605	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
70	2004-08-03 12:00:19.219643	10.126.10.170	10.126.10.170	ICMP	Echo (ping) request
71	2004-08-03 12:00:19.219661	10.126.10.170	10.126.10.170	ICMP	Echo (ping) reply

Gambar 4.11 Hasil Ethereal 10.126.10.169 Ping 10.126.10.170 (1.1)

Berdasarkan gambar 4.11, Router 10.126.10.170 kembali dapat terhubung setelah 9.000013 detik.

No. -	Time	Source	Destination	Protocol	Info
4	2004-08-03 12:00:10.219594	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
8	2004-08-03 12:00:11.219612	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
12	2004-08-03 12:00:12.219562	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
17	2004-08-03 12:00:13.219509	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
42	2004-08-03 12:00:17.219570	10.126.100.1	10.126.100.1	ICMP	Echo (ping) request
43	2004-08-03 12:00:17.219646	10.126.100.1	10.126.100.1	ICMP	Echo (ping) reply

Gambar 4.12 Hasil Ethereal 10.126.100.5 Ping 10.126.100.1 (1.1)

Berdasarkan gambar 4.12, Router 10.126.100.1 kembali dapat terhubung setelah 7.000052 detik.

Gambar 4.13 dan 4.14 merupakan hasil ethereal pada router darqueenice.

No. -	Time	Source	Destination	Protocol	Info
2	2004-08-03 11:59:57.895285	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
3	2004-08-03 11:59:57.895318	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply
48	2004-08-03 12:00:06.759517	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
49	2004-08-03 12:00:06.759522	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
50	2004-08-03 12:00:06.759548	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
51	2004-08-03 12:00:06.759689	10.126.100.1	10.126.100.6	ICMP	Echo (ping) reply

Gambar 4.13 Hasil Ethereal 10.126.100.6 Ping 10.126.100.1 (1.1)

Berdasarkan gambar 4.13, Router 10.126.100.1 kembali dapat terhubung setelah 0.000172 detik.

No. -	Time	Source	Destination	Protocol	Info
53	2004-08-03 12:00:08.209654	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
54	2004-08-03 12:00:08.209823	10.126.10.170	10.126.10.169	ICMP	Echo (ping) reply

Gambar 4.14 Hasil Ethereal 10.126.10.169 Ping 10.126.10.170 (1.1)

Berdasarkan gambar 4.14, Router 10.126.10.170 kembali dapat terhubung setelah 0.000169 detik.

2. Mengirim File

Router aktif adalah darqueenice sedangkan router standby adalah falcon dan hsrp. Klien yang digunakan adalah 10.126.10.205 dan 10.126.100.3. Gambar 4.15, 4.16 dan 4.17 merupakan hasil ethereal pada klien 10.126.10.205

No. -	Time	Source	Destination	Protocol	Info
6	2004-08-03 12:07:19.906884	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
15	2004-08-03 12:07:25.172947	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
26	2004-08-03 12:07:30.673913	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
31	2004-08-03 12:07:30.674536	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.15 Hasil Ethereal 10.126.10.205 Ping 10.126.100.1 (1.2)

Berdasarkan gambar 4.15, Router 10.126.100.1 kembali dapat terhubung setelah 10.767652 detik.

No. -	Time	Source	Destination	Protocol	Info
5	2004-08-03 12:07:19.942448	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
12	2004-08-03 12:07:25.172755	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
25	2004-08-03 12:07:30.674172	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
28	2004-08-03 12:07:30.674499	10.126.10.170	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.16 Hasil Ethereal 10.126.10.205 Ping 10.126.10.170 (1.2)

Berdasarkan gambar 4.16, Router 10.126.10.170 kembali dapat terhubung setelah 10.732051 detik.

No. -	Time	Source	Destination	Protocol	Info
5	2004-08-03 12:07:19.177	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply
8	2004-08-03 12:07:20.172	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
17	2004-08-03 12:07:25.674	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
33	2004-08-03 12:07:31.173	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
34	2004-08-03 12:07:31.173	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.17 Hasil Ethereal 10.126.10.205 Ping 10.126.100.3 (1.2)

Berdasarkan gambar 4.17, klien 10.126.100.3 kembali dapat terhubung setelah 11.001044 detik.

Gambar 4.17 dan 4.18 merupakan hasil ethereal pada router falcon.

No.	Time	Source	Destination	Protocol	Info
5	2004-08-03 12:15:28.919	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
9	2004-08-03 12:15:29.919	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
13	2004-08-03 12:15:30.919	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
17	2004-08-03 12:15:31.919	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
21	2004-08-03 12:15:32.919	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
37	2004-08-03 12:15:36.919	10.126.100.1	10.126.100.1	ICMP	Echo (ping) request
38	2004-08-03 12:15:36.919	10.126.100.1	10.126.100.1	ICMP	Echo (ping) reply

Gambar 4.18 Hasil Ethereal 10.126.100.5 Ping 10.126.100.1 (1.2)

Berdasarkan gambar 4.18, Router 10.126.100.1 kembali dapat terhubung setelah 8.000055 detik.

No.	Time	Source	Destination	Protocol	Info
5	2004-08-03 12:15:28.259	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
9	2004-08-03 12:15:29.259	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
13	2004-08-03 12:15:30.259	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
17	2004-08-03 12:15:31.259	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
21	2004-08-03 12:15:32.259	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
25	2004-08-03 12:15:33.259	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
29	2004-08-03 12:15:34.259	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
33	2004-08-03 12:15:35.259	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
69	2004-08-03 12:15:39.259	10.126.10.170	10.126.10.170	ICMP	Echo (ping) request
70	2004-08-03 12:15:39.259	10.126.10.170	10.126.10.170	ICMP	Echo (ping) reply

Gambar 4.19 Hasil Ethereal 10.126.10.168 Ping 10.126.10.170 (1.2)

Berdasarkan gambar 4.19, Router 10.126.10.170 kembali dapat terhubung setelah 11.000054 detik.

Gambar 4.19 dan 4.20 merupakan hasil ethereal pada router darqueenice.

No.	Time	Source	Destination	Protocol	Info
36	2004-08-03 12:15:26.129590	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
37	2004-08-03 12:15:26.129754	10.126.100.1	10.126.100.6	ICMP	Echo (ping) reply

Gambar 4.20 Hasil Ethereal 10.126.100.6 Ping 10.126.100.1 (1.2)

Berdasarkan gambar 4.20, Router 10.126.100.1 kembali dapat terhubung setelah 0.000169 detik.

No.	Time	Source	Destination	Protocol	Info
37	2004-08-03 12:15:27.929902	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
38	2004-08-03 12:15:27.929907	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
41	2004-08-03 12:15:27.930168	10.126.10.170	10.126.10.169	ICMP	Echo (ping) reply

Gambar 4.21 Hasil Ethereal 10.126.10.169 Ping 10.126.10.170 (1.2)

Berdasarkan gambar 4.21, Router 10.126.10.170 kembali dapat terhubung setelah 0.000266 detik.

4.3.2 Uji Coba 2

1. Tanpa mengirim file.

Router aktif adalah falcon sedangkan router standby adalah hsrp. Klien yang digunakan adalah 10.126.10.205 dan 10.126.100.3. Gambar 4.22, 4.23 dan 4.24 merupakan hasil ethereal pada klien 10.126.10.205

No.	Time	Source	Destination	Protocol	Info
26	2004-08-03 11:58:16.146266	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
35	2004-08-03 11:58:21.647645	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
46	2004-08-03 11:58:27.146867	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
49	2004-08-03 11:58:27.147357	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.22 Hasil Ethereal 10.126.10.205 Ping 10.126.100.1 (2.1)

Berdasarkan gambar 4.22, router 10.126.100.1 kembali dapat terhubung setelah 12.000114 detik.

No.	Time	Source	Destination	Protocol	Info
792	2004-08-03 12:10:01.164943	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
803	2004-08-03 12:10:06.180516	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
814	2004-08-03 12:10:11.680868	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
815	2004-08-03 12:10:11.680993	10.126.10.170	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.23 Hasil Ethereal 10.126.10.205 Ping 10.126.10.170 (2.1)

Berdasarkan gambar 4.23, router 10.126.10.170 kembali dapat terhubung setelah 11.937926 detik.

No.	Time	Source	Destination	Protocol	Info
7	2004-08-03 11:58:16.146348	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
16	2004-08-03 11:58:21.647948	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
28	2004-08-03 11:58:27.147345	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
30	2004-08-03 11:58:27.148235	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.24 Hasil Ethereal 10.126.10.205 Ping 10.126.100.3 (2.1)

Berdasarkan gambar 4.24, klien 10.126.100.3 kembali dapat terhubung setelah 12.001034 detik.

Gambar 4.25 dan 4.26 merupakan hasil ethereal pada router hsrp.

No.	Time	Source	Destination	Protocol	Info
5	2004-08-03 12:06:05.651727	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
9	2004-08-03 12:06:06.651740	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
13	2004-08-03 12:06:07.651739	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
17	2004-08-03 12:06:08.651726	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
21	2004-08-03 12:06:09.651726	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
25	2004-08-03 12:06:10.651726	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
29	2004-08-03 12:06:11.651726	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
51	2004-08-03 12:06:15.651726	10.126.100.1	10.126.100.1	ICMP	Echo (ping) request
52	2004-08-03 12:06:15.651735	10.126.100.1	10.126.100.1	ICMP	Echo (ping) reply

Gambar 4.25 Hasil Ethereal 10.126.100.4 Ping 10.126.100.1 (2.1)

Berdasarkan gambar 4.25, Router 10.126.100.1 kembali dapat terhubung setelah 9.999995 detik.

No.	Time	Source	Destination	Protocol	Info
5	2004-08-03 12:17:51.061735	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
10	2004-08-03 12:17:52.061733	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
15	2004-08-03 12:17:53.061733	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
20	2004-08-03 12:17:54.061743	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
25	2004-08-03 12:17:55.061734	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
42	2004-08-03 12:17:59.061746	10.126.10.170	10.126.10.170	ICMP	Echo (ping) request
43	2004-08-03 12:17:59.061766	10.126.10.170	10.126.10.170	ICMP	Echo (ping) reply

Gambar 4.26 Hasil Ethereal 10.126.10.167 Ping 10.126.10.170 (2.1)

Berdasarkan gambar 4.26, Router 10.126.10.170 kembali dapat terhubung setelah 10.000012 detik.

Gambar 4.27 dan 4.28 merupakan hasil ethereal pada router falcon.

No. -	Time	Source	Destination	Protocol	Info
39	2004-08-03 12:06:34.299461	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
40	2004-08-03 12:06:34.299591	10.126.10.170	10.126.10.168	ICMP	Echo (ping) reply

Gambar 4.27 Hasil Ethereal 10.126.10.169 Ping 10.126.10.170 (2.1)

Berdasarkan gambar 4.27, Router 10.126.10.170 kembali dapat terhubung setelah 0.00013 detik.

No. -	Time	Source	Destination	Protocol	Info
47	2004-08-03 12:06:35.269468	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
48	2004-08-03 12:06:35.269476	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
49	2004-08-03 12:06:35.269579	10.126.100.1	10.126.100.5	ICMP	Echo (ping) reply

Gambar 4.28 Hasil Ethereal 10.126.100.5 Ping 10.126.100.1 (2.1)

Berdasarkan gambar 4.28, Router 10.126.100.1 kembali dapat terhubung setelah 0.000111 detik.

2. Mengirim File

Router aktif adalah hsrp sedangkan router standby adalah falcon. Klien yang digunakan adalah 10.126.10.205 dan 10.126.100.3. Gambar 4.29, 4.30 dan 4.31 merupakan hasil ethereal pada klien 10.126.10.205

No. -	Time	Source	Destination	Protocol	Info
791	2004-08-03 12:10:01.195891	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
803	2004-08-03 12:10:06.680717	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
815	2004-08-03 12:10:12.180899	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
816	2004-08-03 12:10:12.181676	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.29 Hasil Ethereal 10.126.10.205 Ping 10.126.100.1 (2.2)

Berdasarkan gambar 4.29, Router 10.126.100.1 kembali dapat terhubung setelah 10.985785 detik.

No. -	Time	Source	Destination	Protocol	Info
792	2004-08-03 12:10:01.164943	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
803	2004-08-03 12:10:06.180516	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
814	2004-08-03 12:10:11.680868	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
815	2004-08-03 12:10:11.680993	10.126.10.170	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.30 Hasil Ethereal 10.126.10.205 Ping 10.126.10.170 (2.2)

Berdasarkan gambar 4.30, Router 10.126.10.170 kembali dapat terhubung setelah 10.922907 detik.

No. -	Time	Source	Destination	Protocol	Info
3	2004-08-03 12:10:00.212376	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply
792	2004-08-03 12:10:01.211518	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
802	2004-08-03 12:10:06.680631	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
814	2004-08-03 12:10:12.180818	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
817	2004-08-03 12:10:12.181752	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.31 Hasil Ethereal 10.126.10.205 Ping 10.126.100.3 (2.2)

Berdasarkan gambar 4.31, klien 10.126.100.3 kembali dapat terhubung setelah 10.970234 detik.

Gambar 4.32 dan 4.33 merupakan hasil ethereal pada router falcon.

No. -	Time	Source	Destination	Protocol	Info
46	2004-08-03 12:18:19.989471	10.126.100.5	10.126.100.1	ICMP	Echo (ping) request
47	2004-08-03 12:18:19.989571	10.126.100.1	10.126.100.5	ICMP	Echo (ping) reply

Gambar 4.32 Hasil Ethereal 10.126.100.5 Ping 10.126.100.1 (2.2)

Berdasarkan gambar 4.32, Router 10.126.100.1 kembali dapat terhubung setelah 0.0001 detik.

No. -	Time	Source	Destination	Protocol	Info
30	2004-08-03 12:18:18.299466	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
31	2004-08-03 12:18:18.299474	10.126.10.168	10.126.10.170	ICMP	Echo (ping) request
32	2004-08-03 12:18:18.299592	10.126.10.170	10.126.10.168	ICMP	Echo (ping) reply

Gambar 4.33 Hasil Ethereal 10.126.10.168 Ping 10.126.10.170 (2.2)

Berdasarkan gambar 4.33, Router 10.126.10.170 kembali dapat terhubung setelah 0.000126 detik.

Gambar 4.34 dan 4.35 merupakan hasil ethereal pada router hsrp.

No.	Time	Source	Destination	Protocol	Info
5	2004-08-03 12:17:50.831776	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
10	2004-08-03 12:17:51.831781	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
15	2004-08-03 12:17:52.831786	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
20	2004-08-03 12:17:53.831810	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
25	2004-08-03 12:17:54.831793	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
30	2004-08-03 12:17:55.831799	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
60	2004-08-03 12:18:00.831734	10.126.100.1	10.126.100.1	ICMP	Echo (ping) request
61	2004-08-03 12:18:00.831750	10.126.100.1	10.126.100.1	ICMP	Echo (ping) reply

Gambar 4.34 Hasil Ethereal 10.126.100.4 Ping 10.126.100.1 (2.2)

Berdasarkan gambar 4.34, Router 10.126.100.1 kembali dapat terhubung setelah 9.999974 detik.

No.	Time	Source	Destination	Protocol	Info
5	2004-08-03 12:17:51.061735	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
10	2004-08-03 12:17:52.061733	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
15	2004-08-03 12:17:53.061733	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
20	2004-08-03 12:17:54.061743	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
25	2004-08-03 12:17:55.061734	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
42	2004-08-03 12:17:59.061746	10.126.10.170	10.126.10.170	ICMP	Echo (ping) request
43	2004-08-03 12:17:59.061766	10.126.10.170	10.126.10.170	ICMP	Echo (ping) reply

Gambar 4.35 Hasil Ethereal 10.126.10.167 Ping 10.126.10.170 (2.2)

Berdasarkan gambar 4.35, Router 10.126.10.170 kembali dapat terhubung setelah 8.000031 detik.

4.3.3 Uji Coba 3

1. Tanpa mengirim file.

Router aktif adalah darqueenice sedangkan router standby adalah hsrp. Klien yang digunakan adalah 10.126.10.205 dan 10.126.100.3. Gambar 4.34, 4.35 dan 4.36 merupakan hasil ethereal pada klien 10.126.10.205

No. -	Time	Source	Destination	Protocol	Info
5	2004-08-03 14:52:15.310721	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply
11	2004-08-03 14:52:16.310138	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
24	2004-08-03 14:52:21.669801	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
47	2004-08-03 14:52:27.154983	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
49	2004-08-03 14:52:27.155054	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.36 Hasil Ethereal 10.126.10.205 Ping 10.126.100.1 (3.1)

Berdasarkan gambar 4.36, router 10.126.100.1 kembali dapat terhubung setelah 10.844916 detik.

No. -	Time	Source	Destination	Protocol	Info
3	2004-08-03 14:52:15.310139	10.126.10.170	10.126.10.205	ICMP	Echo (ping) reply
10	2004-08-03 14:52:16.310062	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
25	2004-08-03 14:52:21.669876	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
42	2004-08-03 14:52:27.154380	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
46	2004-08-03 14:52:27.154562	10.126.10.170	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.37 Hasil Ethereal 10.126.10.205 Ping 10.126.10.170 (3.1)

Berdasarkan gambar 4.37, router 10.126.10.170 kembali dapat terhubung setelah 10.8445 detik.

No. -	Time	Source	Destination	Protocol	Info
8	2004-08-03 14:52:16.153982	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
23	2004-08-03 14:52:21.654107	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
43	2004-08-03 14:52:27.154473	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
48	2004-08-03 14:52:27.155029	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.38 Hasil Ethereal 10.126.10.205 Ping 10.126.100.3 (3.1)

Berdasarkan gambar 4.38, klien 10.126.100.3 kembali dapat terhubung setelah 11.001047 detik.

Gambar 4.39 dan 4.40 merupakan hasil ethereal pada router hsrp.

No. -	Time	Source	Destination	Protocol	Info
6	2004-08-03 15:00:05.831695	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
10	2004-08-03 15:00:06.831695	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
14	2004-08-03 15:00:07.831699	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
18	2004-08-03 15:00:08.831695	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
22	2004-08-03 15:00:09.831695	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
26	2004-08-03 15:00:10.831701	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
30	2004-08-03 15:00:11.831697	10.126.100.4	10.126.100.1	ICMP	Echo (ping) request
62	2004-08-03 15:00:15.831693	10.126.100.1	10.126.100.1	ICMP	Echo (ping) request
63	2004-08-03 15:00:15.831703	10.126.100.1	10.126.100.1	ICMP	Echo (ping) reply

Gambar 4.39 Hasil Ethereal 10.126.100.4 Ping 10.126.100.1 (3.1)

Berdasarkan gambar 4.39, Router 10.126.100.1 kembali dapat terhubung setelah 10.000008 detik.

No. -	Time	Source	Destination	Protocol	Info
5	2004-08-03 15:00:05.931687	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
9	2004-08-03 15:00:06.931687	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
13	2004-08-03 15:00:07.931712	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
17	2004-08-03 15:00:08.931719	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
21	2004-08-03 15:00:09.931688	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
44	2004-08-03 15:00:13.931689	10.126.10.170	10.126.10.170	ICMP	Echo (ping) request
45	2004-08-03 15:00:13.931701	10.126.10.170	10.126.10.170	ICMP	Echo (ping) reply

Gambar 4.40 Hasil Ethereal 10.126.10.167 Ping 10.126.10.170 (3.1)

Berdasarkan gambar 4.40, Router 10.126.10.170 kembali dapat terhubung setelah 8.000014 detik.

Gambar 4.41 dan 4.42 merupakan hasil ethereal pada router darqueenice.

No. -	Time	Source	Destination	Protocol	Info
25	2004-08-03 15:00:22.291997	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
26	2004-08-03 15:00:22.292002	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
27	2004-08-03 15:00:22.292093	10.126.10.170	10.126.10.169	ICMP	Echo (ping) reply

Gambar 4.41 Hasil Ethereal 10.126.10.168 Ping 10.126.10.170 (3.1)

Berdasarkan gambar 4.41, Router 10.126.10.170 kembali dapat terhubung setelah 0.000096 detik.

No. -	Time	Source	Destination	Protocol	Info
48	2004-08-03 15:00:24.212015	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
49	2004-08-03 15:00:24.212106	10.126.100.1	10.126.100.6	ICMP	Echo (ping) reply

Gambar 4.42 Hasil Ethereal 10.126.100.6 Ping 10.126.100.1 (3.1)

Berdasarkan gambar 4.42, Router 10.126.100.1 kembali dapat terhubung setelah 0.000091 detik.

2. Mengirim File

Router aktif adalah hsrp sedangkan router standby adalah darqueenice. Klien yang digunakan adalah 10.126.10.205 dan 10.126.100.3. Gambar 4.41, 4.42 dan 4.43 merupakan hasil ethereal pada klien 10.126.10.205

No. -	Time	Source	Destination	Protocol	Info
17	2004-08-03 14:55:49.226870	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply
527	2004-08-03 14:55:50.226803	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
542	2004-08-03 14:55:55.664631	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
558	2004-08-03 14:56:01.164920	10.126.10.205	10.126.100.1	ICMP	Echo (ping) request
564	2004-08-03 14:56:01.165076	10.126.100.1	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.43 Hasil Ethereal 10.126.10.205 Ping 10.126.100.1 (3.2)

Berdasarkan gambar 4.43, Router 10.126.100.1 kembali dapat terhubung setelah 10.938273 detik.

No. -	Time	Source	Destination	Protocol	Info
526	2004-08-03 14:55:50.195716	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
541	2004-08-03 14:55:55.664537	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
557	2004-08-03 14:56:01.164836	10.126.10.205	10.126.10.170	ICMP	Echo (ping) request
563	2004-08-03 14:56:01.165041	10.126.10.170	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.44 Hasil Ethereal 10.126.10.205 Ping 10.126.10.170 (3.2)

Berdasarkan gambar 4.44, Router 10.126.10.170 kembali dapat terhubung setelah 10.969325 detik.

No.	Time	Source	Destination	Protocol	Info
180	2004-08-03 14:55:49.372133	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply
528	2004-08-03 14:55:50.367544	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
543	2004-08-03 14:55:55.664699	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
569	2004-08-03 14:56:01.165369	10.126.10.205	10.126.100.3	ICMP	Echo (ping) request
570	2004-08-03 14:56:01.165843	10.126.100.3	10.126.10.205	ICMP	Echo (ping) reply

Gambar 4.45 Hasil Ethereal 10.126.10.205 Ping 10.126.100.3 (3.2)

Berdasarkan gambar 4.45, klien 10.126.100.3 kembali dapat terhubung setelah 11.793236 detik.

Gambar 4.46 dan 4.47 merupakan hasil ethereal pada router darqueenice.

No.	Time	Source	Destination	Protocol	Info
5	2004-08-03 15:03:48.671944	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
9	2004-08-03 15:03:49.671942	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
13	2004-08-03 15:03:50.671944	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
17	2004-08-03 15:03:51.672130	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
21	2004-08-03 15:03:52.671946	10.126.100.6	10.126.100.1	ICMP	Echo (ping) request
37	2004-08-03 15:03:56.671953	10.126.100.1	10.126.100.1	ICMP	Echo (ping) request
38	2004-08-03 15:03:56.671986	10.126.100.1	10.126.100.1	ICMP	Echo (ping) reply

Gambar 4.46 Hasil Ethereal 10.126.100.6 Ping 10.126.100.1 (3.2)

Berdasarkan gambar 4.46, Router 10.126.100.1 kembali dapat terhubung setelah 8.000042 detik.

No.	Time	Source	Destination	Protocol	Info
4	2004-08-03 15:03:48.791937	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
8	2004-08-03 15:03:49.791936	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
12	2004-08-03 15:03:50.791942	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
16	2004-08-03 15:03:51.791937	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
20	2004-08-03 15:03:52.791936	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
24	2004-08-03 15:03:53.791949	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
28	2004-08-03 15:03:54.791936	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
32	2004-08-03 15:03:55.791948	10.126.10.169	10.126.10.170	ICMP	Echo (ping) request
69	2004-08-03 15:03:59.791969	10.126.10.170	10.126.10.170	ICMP	Echo (ping) request
70	2004-08-03 15:03:59.792002	10.126.10.170	10.126.10.170	ICMP	Echo (ping) reply

Gambar 4.47 Hasil Ethereal 10.126.10.169 Ping 10.126.10.170 (3.2)

Berdasarkan gambar 4.47, Router 10.126.10.170 kembali dapat terhubung setelah 11.0009137 detik.

Gambar 4.48 dan 4.49 merupakan hasil ethereal pada router hsrp.

No. -	Time	Source	Destination	Protocol	Info
2	2004-08-03 15:03:48.671943	10.126.100.1	10.126.100.4	ICMP	Echo (ping) reply

Gambar 4.48 Hasil Ethereal 10.126.100.4 Ping 10.126.100.1 (3.2)

Berdasarkan gambar 4.48, Router 10.126.100.1 kembali dapat terhubung setelah 0.000111 detik.

No. -	Time	Source	Destination	Protocol	Info
2	2004-08-03 15:03:50.737047	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
5	2004-08-03 15:03:50.737216	10.126.10.170	10.126.10.167	ICMP	Echo (ping) reply

Gambar 4.49 Hasil Ethereal 10.126.10.167 Ping 10.126.10.170 (3.2)

Berdasarkan gambar 4.49, Router 10.126.10.170 kembali dapat terhubung setelah 0.000175 detik.

4.3.4 Uji Coba 4

Router tiga merupakan router aktif, sedangkan router satu, dua dan empat merupakan router standby. Berikut hasil pencatatan waktu syslog sebelum router tiga dimatikan.

```
Aug 11 08:36:56 umlDasamuka insco[166]: reg 1
Aug 11 08:36:59 umlDasamuka insco[166]: terima S...
Aug 11 08:36:59 umlDasamuka insco[166]: terima A...
Aug 11 08:37:02 umlDasamuka insco[166]: reg 1
Aug 11 08:37:05 umlDasamuka insco[166]: terima S...
Aug 11 08:37:05 umlDasamuka insco[166]: terima A...
Aug 11 08:37:09 umlDasamuka insco[166]: reg 1
Aug 11 08:37:11 umlDasamuka insco[166]: terima S...
Aug 11 08:37:11 umlDasamuka insco[166]: terima A...
Aug 11 08:37:15 umlDasamuka insco[166]: reg 1
```

Gambar 4. 50 Router Satu Sebelum Router Aktif Mati (4)

Berdasarkan gambar 4.50 router satu akan menerima pesan 'A' setelah 3 detik dari awal waktu standby.

```
Aug 11 08:28:32 umlDasamuka insco[164]: reg 1
Aug 11 08:28:38 umlDasamuka insco[164]: terima S...
Aug 11 08:28:38 umlDasamuka insco[164]: terima A...
Aug 11 08:28:38 umlDasamuka insco[164]: reg 1
Aug 11 08:28:44 umlDasamuka insco[164]: terima S...
Aug 11 08:28:44 umlDasamuka insco[164]: terima A...
Aug 11 08:28:44 umlDasamuka insco[164]: reg 1
```

Gambar 4. 51 Router Dua Sebelum Router Aktif Mati (4)

Berdasarkan gambar 4.51 router Dua akan menerima pesan 'A' setelah 6 detik dari awal waktu standby.

```
Aug 11 08:28:50 umlDasamuka insco[158]: reg 1
Aug 11 08:28:56 umlDasamuka insco[158]: reg 1 jum 0
Aug 11 08:28:56 umlDasamuka insco[158]: reg 2
Aug 11 08:28:56 umlDasamuka insco[158]: kirim ...
Aug 11 08:28:56 umlDasamuka insco[158]: terima S...
Aug 11 08:28:56 umlDasamuka insco[158]: terima A...
Aug 11 08:28:56 umlDasamuka insco[158]: reg 1
```

Gambar 4. 52 Router Empat Sebelum Router Aktif Mati (4)

Berdasarkan gambar 4.52 router Empat akan menerima pesan 'A' bersamaan waktu standby.

Berikut hasil pencatatan waktu syslog setelah router tiga dimatikan :

```
Aug 11 08:41:28 umlDasamuka insco[166]: Pemilihan Router
Aug 11 08:41:28 umlDasamuka insco[166]: ioctl (SIOCSFLAGS)
Cannot assign requested address
Aug 11 08:41:28 umlDasamuka insco[166]: downeth1 DOWN eth0:0
Aug 11 08:41:28 umlDasamuka insco[166]: downeth1 DOWN eth0
Aug 11 08:41:28 umlDasamuka insco[166]: NOW UP eth0
Aug 11 08:41:28 umlDasamuka insco[166]: restore.........
Aug 11 08:41:28 umlDasamuka insco[166]: Menjadi standby
Aug 11 08:41:28 umlDasamuka insco[166]: ip 10 126 1 19
Aug 11 08:41:28 umlDasamuka insco[166]: prio 5 ip 19
Aug 11 08:41:31 umlDasamuka insco[166]: prio 5 ip 19
Aug 11 08:41:31 umlDasamuka insco[166]: Menjadi aktif
```

Gambar 4. 53 Router Satu Setelah Router Aktif Mati (4)

```
Aug 11 08:41:26 umlDasamuka insco[149]: ip 10 126 1 20
Aug 11 08:41:26 umlDasamuka insco[149]: prio 4 ip 20
Aug 11 08:41:28 umlDasamuka insco[149]: ip 10 126 1 19
Aug 11 08:41:28 umlDasamuka insco[149]: prio 5 ip 19
Aug 11 08:41:29 umlDasamuka insco[149]: prio 5 ip 20
Aug 11 08:41:29 umlDasamuka insco[149]: Menjadi standby
Aug 11 08:41:29 umlDasamuka insco[149]: ioctl (SIOCSFLAGS)
Cannot assign requested address
Aug 11 08:41:29 umlDasamuka insco[149]: downeth1 DOWN eth0:0
Aug 11 08:41:29 umlDasamuka insco[149]: downeth1 DOWN eth0
Aug 11 08:41:29 umlDasamuka insco[149]: NOW UP eth0
Aug 11 08:41:29 umlDasamuka insco[149]: restore.........
Aug 11 08:41:29 umlDasamuka insco[149]: Menjadi standby
```

Gambar 4. 54 Router Dua Setelah Router Aktif Mati (4)

Berdasarkan gambar 4.53 router satu menjadi router aktif setelah proses pemilihan router. Pada gambar 4.54 router dua memiliki prioritas lebih rendah daripada router satu sehingga menjadi standby.

```
Aug 11 08:41:24 umlDasamuka insco[168]: ip 10 126 1 22
Aug 11 08:41:24 umlDasamuka insco[168]: prio 2 ip 22
Aug 11 08:41:26 umlDasamuka insco[168]: ip 10 126 1 20
Aug 11 08:41:26 umlDasamuka insco[168]: prio 5 ip 20
Aug 11 08:41:27 umlDasamuka insco[168]: prio 3 ip 22
Aug 11 08:41:27 umlDasamuka insco[168]: Menjadi standby
Aug 11 08:41:27 umlDasamuka insco[168]: ioctl (SIOCSFLAGS)
Cannot assign requested address
Aug 11 08:41:27 umlDasamuka insco[168]: downeth1 DOWN eth0:0
Aug 11 08:41:27 umlDasamuka insco[168]: downeth1 DOWN eth0
Aug 11 08:41:27 umlDasamuka insco[168]: NOW UP eth0
Aug 11 08:41:27 umlDasamuka insco[168]: restore.........
Aug 11 08:41:27 umlDasamuka insco[168]: Menjadi standby
```

Gambar 4. 55 Router Empat Setelah Router Aktif Mati (4)

Berdasarkan gambar 4.55 router empat memiliki prioritas lebih rendah daripada router dua sehingga menjadi standby.

```
Aug 11 08:41:19 umlDasamuka insco[152]: restore.........
Aug 11 08:41:19 umlDasamuka insco[152]: Menjadi standby
Aug 11 08:41:19 umlDasamuka insco[152]: Keluar 15
```

Gambar 4. 56 Router Empat Mati (4)

Berdasarkan gambar 4.56 dan 4.53 perpindahan router dari router tiga ke router satu berlangsung selama 12 detik.

4.3.5 Uji Coba 5

Router satu merupakan router aktif, sedangkan router dua, tiga, empat dan lima merupakan router standby. Berikut hasil pencatatan waktu syslog sebelum router satu dimatikan

```
Aug 11 08:47:13 umlDasamuka insco[149]: reg 1
Aug 11 08:47:16 umlDasamuka insco[149]: terima S...
Aug 11 08:47:16 umlDasamuka insco[149]: terima A...
Aug 11 08:47:19 umlDasamuka insco[149]: reg 1
Aug 11 08:47:22 umlDasamuka insco[149]: terima S...
Aug 11 08:47:22 umlDasamuka insco[149]: terima A...
Aug 11 08:47:25 umlDasamuka insco[149]: reg 1
Aug 11 08:47:28 umlDasamuka insco[149]: terima S...
Aug 11 08:47:28 umlDasamuka insco[149]: terima A...
Aug 11 08:47:31 umlDasamuka insco[149]: reg 1
```

Gambar 4. 57 Router Dua Sebelum Router Aktif Mati (5)

Berdasarkan gambar 4.57 router Dua akan menerima pesan 'A' setelah 3 detik dari awal waktu *standby*.

```
Aug 11 08:47:28 umlDasamuka insco[159]: reg 1
Aug 11 08:47:35 umlDasamuka insco[159]: reg 1 jum 0
Aug 11 08:47:35 umlDasamuka insco[159]: reg 2
Aug 11 08:47:35 umlDasamuka insco[159]: kirim ...
Aug 11 08:47:35 umlDasamuka insco[159]: terima S...
Aug 11 08:47:35 umlDasamuka insco[159]: terima A...
Aug 11 08:47:35 umlDasamuka insco[159]: reg 1
Aug 11 08:47:41 umlDasamuka insco[159]: reg 1 jum 0
Aug 11 08:47:41 umlDasamuka insco[159]: reg 2
Aug 11 08:47:41 umlDasamuka insco[159]: kirim ...
Aug 11 08:47:41 umlDasamuka insco[159]: terima S...
Aug 11 08:47:41 umlDasamuka insco[159]: terima A...
Aug 11 08:47:41 umlDasamuka insco[159]: reg 1
```

Gambar 4. 58 Router Tiga Sebelum Router Aktif Mati (5)

Berdasarkan gambar 4.58 router Tiga akan menerima pesan 'A' bersamaan waktu *standby*.

```
Aug 11 08:47:04 umlDasamuka insco[168]: reg 1
Aug 11 08:47:08 umlDasamuka insco[168]: terima R...
Aug 11 08:47:08 umlDasamuka insco[168]: terima A...
Aug 11 08:47:11 umlDasamuka insco[168]: reg 1
Aug 11 08:47:16 umlDasamuka insco[168]: terima S...
Aug 11 08:47:16 umlDasamuka insco[168]: terima A...
Aug 11 08:47:17 umlDasamuka insco[168]: reg 1
Aug 11 08:47:22 umlDasamuka insco[168]: terima S...
Aug 11 08:47:22 umlDasamuka insco[168]: terima A...
Aug 11 08:47:23 umlDasamuka insco[168]: reg 1
```

Gambar 4. 59 Router Empat Sebelum Router Aktif Mati (5)

Berdasarkan gambar 4.59 router Empat akan menerima pesan 'A' setelah 5 detik dari awal waktu standby.

```
Aug 11 08:47:18 umlDasamuka insco[155]: reg 1
Aug 11 08:47:22 umlDasamuka insco[155]: terima S...
Aug 11 08:47:22 umlDasamuka insco[155]: terima A...
Aug 11 08:47:24 umlDasamuka insco[155]: reg 1
Aug 11 08:47:28 umlDasamuka insco[155]: terima S...
Aug 11 08:47:28 umlDasamuka insco[155]: terima A...
Aug 11 08:47:30 umlDasamuka insco[155]: reg 1
Aug 11 08:47:35 umlDasamuka insco[155]: terima S...
Aug 11 08:47:35 umlDasamuka insco[155]: terima A...
Aug 11 08:47:36 umlDasamuka insco[155]: reg 1
```

Gambar 4. 60 Router Lima Sebelum Router Aktif Mati (5)

Berdasarkan gambar 4.60 router Lima akan menerima pesan 'A' setelah 4 detik dari awal waktu standby.

Berikut hasil pencatatan waktu syslog setelah router satu dimatikan :

```
Aug 11 08:51:01 umlDasamuka insco[149]: Pemilihan Router
Aug 11 08:51:01 umlDasamuka insco[149]: ioctl (SIOCSFLAGS)
Cannot assign requested address
Aug 11 08:51:01 umlDasamuka insco[149]: downeth1 DOWN eth0:0
Aug 11 08:51:01 umlDasamuka insco[149]: downeth1 DOWN eth0
Aug 11 08:51:01 umlDasamuka insco[149]: NOW UP eth0
Aug 11 08:51:01 umlDasamuka insco[149]: restore.........
Aug 11 08:51:01 umlDasamuka insco[149]: Menjadi standby
Aug 11 08:51:01 umlDasamuka insco[149]: ip 10 126 1 20
Aug 11 08:51:01 umlDasamuka insco[149]: prio 3 ip 20
Aug 11 08:51:04 umlDasamuka insco[149]: prio 3 ip 20
Aug 11 08:51:04 umlDasamuka insco[149]: Menjadi aktif
```

Gambar 4. 61 Router Dua Setelah Router Aktif Mati (5)

Berdasarkan gambar 4.61 router dua menjadi router aktif setelah proses pemilihan router. Pada gambar 4.62 router tiga memiliki prioritas lebih rendah daripada router dua sehingga menjadi standby. Sedangkan pada gambar 4.63 router empat memiliki prioritas lebih rendah daripada router dua sehingga menjadi standby.

```
Aug 11 08:50:58 umlDasamuka insco[159]: Pemilihan Router
Aug 11 08:50:58 umlDasamuka insco[159]: ioctl (SIOCSFLAGS)
Cannot assign requested address
Aug 11 08:50:58 umlDasamuka insco[159]: downeth1 DOWN eth0:0
Aug 11 08:50:58 umlDasamuka insco[159]: downeth1 DOWN eth0
Aug 11 08:50:58 umlDasamuka insco[159]: NOW UP eth0
Aug 11 08:50:58 umlDasamuka insco[159]: restore.........
Aug 11 08:50:58 umlDasamuka insco[159]: Menjadi standby
Aug 11 08:50:58 umlDasamuka insco[159]: ip 10 126 1 21
Aug 11 08:50:58 umlDasamuka insco[159]: prio 2 ip 21
Aug 11 08:50:58 umlDasamuka insco[159]: ip 10 126 1 22
Aug 11 08:50:58 umlDasamuka insco[159]: prio 2 ip 22
Aug 11 08:50:59 umlDasamuka insco[159]: ip 10 126 1 23
Aug 11 08:50:59 umlDasamuka insco[159]: prio 1 ip 23
Aug 11 08:51:01 umlDasamuka insco[159]: ip 10 126 1 20
Aug 11 08:51:01 umlDasamuka insco[159]: prio 3 ip 20
Aug 11 08:51:01 umlDasamuka insco[159]: reg 1
Aug 11 08:51:04 umlDasamuka insco[159]: terima A...
```

Gambar 4. 62 Router Tiga Setelah Router Aktif Mati (5)

```
Aug 11 08:50:58 umlDasamuka insco[168]: Pemilihan Router
Aug 11 08:50:58 umlDasamuka insco[168]: ioctl (SIOCSFLAGS)
Cannot assign requested address
Aug 11 08:50:58 umlDasamuka insco[168]: downeth1 DOWN eth0:0
Aug 11 08:50:58 umlDasamuka insco[168]: downeth1 DOWN eth0
Aug 11 08:50:58 umlDasamuka insco[168]: NOW UP eth0
Aug 11 08:50:58 umlDasamuka insco[168]: restore.........
Aug 11 08:50:58 umlDasamuka insco[168]: Menjadi standby
Aug 11 08:50:59 umlDasamuka insco[168]: ip 10 126 1 22
Aug 11 08:50:59 umlDasamuka insco[168]: prio 2 ip 22
Aug 11 08:50:59 umlDasamuka insco[168]: ip 10 126 1 23
Aug 11 08:50:59 umlDasamuka insco[168]: prio 1 ip 23
Aug 11 08:51:01 umlDasamuka insco[168]: ip 10 126 1 20
Aug 11 08:51:01 umlDasamuka insco[168]: prio 3 ip 20
Aug 11 08:51:02 umlDasamuka insco[168]: prio 3 ip 23
Aug 11 08:51:02 umlDasamuka insco[168]: Menjadi standby
```

Gambar 4. 63 Router Empat Setelah Router Aktif Mati (5)

```
Aug 11 08:50:59 umlDasamuka insco[155]: Pemilihan Router
Aug 11 08:50:59 umlDasamuka insco[155]: ioctl (SIOCSFLAGS)
Cannot assign requested address
Aug 11 08:50:59 umlDasamuka insco[155]: downeth1 DOWN eth0:0
Aug 11 08:50:59 umlDasamuka insco[155]: downeth1 DOWN eth0
Aug 11 08:50:59 umlDasamuka insco[155]: NOW UP eth0
Aug 11 08:50:59 umlDasamuka insco[155]: restore.........
Aug 11 08:50:59 umlDasamuka insco[155]: Menjadi standby
Aug 11 08:50:59 umlDasamuka insco[155]: ip 10 126 1 23
Aug 11 08:50:59 umlDasamuka insco[155]: prio 1 ip 23
Aug 11 08:51:01 umlDasamuka insco[155]: ip 10 126 1 20
Aug 11 08:51:01 umlDasamuka insco[155]: prio 3 ip 20
Aug 11 08:51:03 umlDasamuka insco[155]: prio 3 ip 23
Aug 11 08:51:03 umlDasamuka insco[155]: Menjadi standby
```

Gambar 4. 64 Router Lima Setelah Router Aktif Mati (5)

Berdasarkan gambar 4.64 router lima memiliki prioritas lebih rendah daripada router dua sehingga menjadi *standby*.

```
Aug 11 08:50:55 umlDasamuka insco[166]: restore.........
Aug 11 08:50:55 umlDasamuka insco[166]: Menjadi standby
Aug 11 08:50:55 umlDasamuka insco[166]: Keluar 15
```

Gambar 4. 65 Router Satu Mati (5)

Berdasarkan gambar 4.65 dan 4.61 perpindahan router dari router satu ke router dua berlangsung selama 9 detik.

4.4 Evaluasi Hasil Uji Coba

Uji coba diatas bertujuan untuk mencatat waktu reply ketika perpindahan router terjadi baik antar klien dengan klien, router dengan klien maupun router dengan router. Tabel 4.2, 4.3, 4.4 dan 4.5 merupakan hasil uji coba dimana hasil perhitungan waktu dalam satuan detik. Untuk tabel 4.2, 4.3 dan 4.4 menggunakan ethereal, sedangkan tabel 4.5 menggunakan waktu yang tertera pada *syslog*.

Tabel 4. 2 Hasil Uji Coba Reply Klien ke Klien

Uji Coba Dari 10.126.10.205	Tujuan(dalam detik) 10.126.100.3
1 tanpa file	11.001283
1 dengan file	11.001044
2 tanpa file	12.001034
2 dengan file	10.970234
2 tanpa file	11.001047
2 dengan file	11.793236

Tabel 4. 3 Hasil Uji Coba Reply Router ke Klien

Uji Coba Dari 10.126.10.205	Tujuan (dalam detik)	
	10.126.100.1	10.126.10.170
1 tanpa file	11.001041	10.922907
1 dengan file	10.767652	10.732051
2 tanpa file	12.000114	11.937926
2 dengan file	10.985785	10.922907
2 tanpa file	10.844916	10.8445
2 dengan file	10.938273	10.969325

Tabel 4. 4 Hasil Uji Coba Reply Router ke router

Uji Coba	Dari	Tujuan (dalam detik)	
		10.126.100.1	10.126.10.170
1 tanpa file	10.126.10.168	7.000052	9.000013
	10.126.10.167	8.000142	10.000268
	10.126.10.169	0.000172	0.000169
1 dengan file	10.126.10.168	8.000055	11.000054
	10.126.10.167	-	-
	10.126.10.169	0.000169	0.000266

2 tanpa file	10.126.10.168	0.000111	0.00013
	10.126.10.167	9.999995	10.000012
	10.126.10.169	-	-
2 dengan file	10.126.10.168	0.0001	0.000126
	10.126.10.167	9.999974	8.000031
	10.126.10.169	-	-
2 tanpa file	10.126.10.168	-	-
	10.126.10.167	10.000008	8.000014
	10.126.10.169	0.000091	0.000096
2 dengan file	10.126.10.168	-	-
	10.126.10.167	0.000111	0.000175
	10.126.10.169	8.000042	11.0009137

Tabel 4. 5 Hasil Uji Coba 4 dan 5 pada Perpindahan Router

Uji Coba	Waktu Perpindahan	Keterangan
4	12 detik	Router tiga ke router satu
5	9 detik	Router satu ke router dua

Berdasarkan tabel hasil uji 4.2, 4.3, 4.4 dan 4.5 kecepatan *reply* antar klien ke klien, router ke klien maupun router ke router ketika terjadi perpindahan router untuk aplikasi Insco tidak dipengaruhi oleh :

1. Lalu lintas jaringan

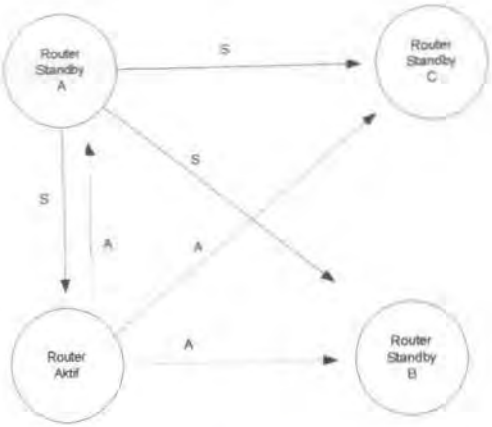
Uji coba satu dan dua berada pada jaringan dengan lalu lintas yang padat daripada uji coba tiga, namun kecepatan *reply* tidak ada perubahan yang signifikan. Selain itu pada uji satu, dua dan tiga ketika terjadi pengiriman file juga tidak mempengaruhi kecepatan *reply*.

2. Jumlah router dalam satu kelompok

Ketika router aktif masih hidup, hanya ada satu router standby yang mengirimkan pesan 'S' secara *broadcast*, sedangkan router standby yang lain menunggu pesan 'A' yang merupakan balasan router aktif terhadap pesan 'S' dari router standby tersebut. Hal ini dilakukan untuk mengurangi lalu lintas jaringan sehingga mempercepat pemilihan router. Pada gambar 4.66 router standby pengirim pesan 'S' adalah router A, sedangkan router penunggu pesan 'A' adalah router B dan C. Router A menunggu selama 6 detik jika tidak menerima 'A' akan melakukan pengiriman pesan 'S', sedangkan router B dan C akan menunggu pesan 'A' selama 6 detik juga. Jika router B dan C tidak menerima pesan 'A' dalam 6 detik, router ini akan melakukan pengiriman pesan 'S', jika dalam 3 detik tidak menerima 'A' maka dilakukan pemilihan router.

Jika router aktif masih hidup, maka router A akan melakukan pengiriman setiap 7 detik karena 6 detik menunggu dan 1 detik mengirim pesan 'S' dan menerima 'A'. Sedangkan router B dan C mengalami setiap 6 detik penungguan pesan 'A'.

Dengan demikian yang pertama kali memasuki tahap pemilihan router adalah router A (gambar 4.67), sedangkan antara router B dan C yang terlebih dahulu akan memasuki tahap ini tergantung dari permulaan waktu standby mereka. Waktu tahap pemilihan router masing-masing router adalah 3 detik selama router tersebut tidak menerima pesan 'A' dari router lain.



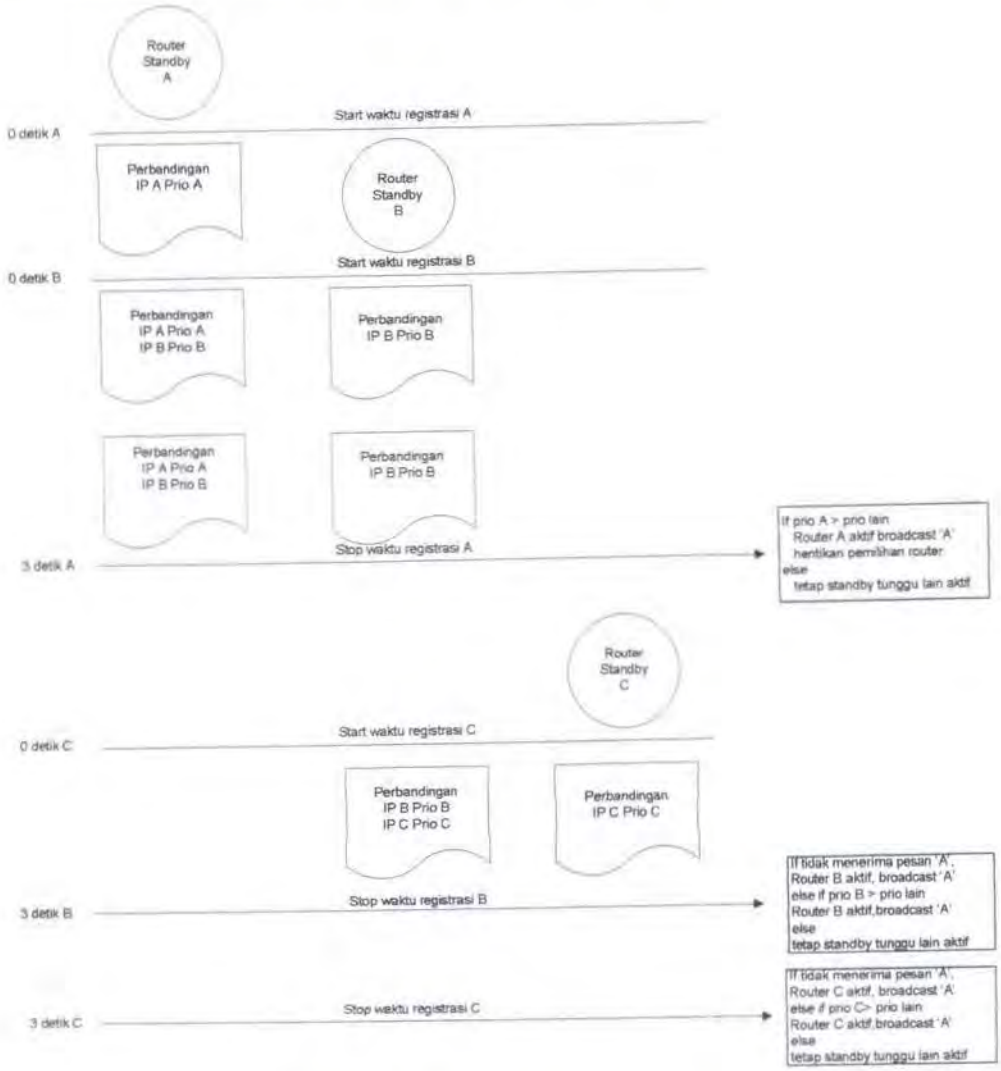
Gambar 4.66 Kondisi Router Aktif Hidup

Ketika router A telah mengalami tahap registrasi, waktu standby router B dan C akan berkurang, setelah itu router B dan C mengirimkan pesan 'S' dan menunggu 3 detik, kemudian memasuki tahap registrasi. Jika waktu standby router B dan C 3 detik, ketika router A memasuki tahap registrasi maka router A akan menjadi router aktif, karena router B dan C belum memasuki tahap registrasi.

Namun jika router B dan C tidak dalam penungguan dan telah mengirimkan pesan 'S' dan menunggu pesan 'A' maka router B dan C akan memasuki tahap pemilihan router. (gambar 4.67)

Sehingga waktu terlama untuk perpindahan router pada kelompok router berjumlah dua adalah 6 detik penungguan pesan 'A' ditambah 3 detik penungguan pesan 'A' setelah pengiriman pesan 'S' ditambah 3 detik tahap registrasi sama dengan 12 detik. Sedangkan waktu terlama untuk perpindahan router pada kelompok router berjumlah lebih dua adalah 5 detik penungguan pesan 'A' (asumsi awal waktu router standby pengirim merupakan akhir waktu standby router penunggu), ditambah 3 detik penungguan pesan 'A' setelah pengiriman

pesan 'S', ditambah 3 detik tahap registrasi, ditambah 2,9999 detik tahap registrasi router standby penunggu jika router ini mempunyai prioritas lebih tinggi daripada router standby pengirim sama dengan 12,9999 detik.



Gambar 4.67 Pemilihan Router

Sehingga dapat disimpulkan dari gambar 4.67 dan uji coba 4 dan 5 untuk perpindahan router untuk lebih dari dua buah router dalam satu kelompok router tidak mempengaruhi kecepatan router. Namun untuk kelompok router yang hanya memiliki router aktif dan satu router standby memiliki kekhususan sendiri karena

router standby yang ada bertindak sebagai router penunggu dan pengirim pesan. Selain itu pada tahap registrasi tidak perlu menunggu router standby lain menjadi aktif. Meskipun demikian waktu perpindahan router untuk kelompok lebih dari dua router dapat lebih cepat dari kelompok dua buah router jika waktu mulai standby antar router memiliki jarak yang jauh atau hampir bersamaan (dekat).

Waktu Request disebabkan oleh Broadcast ARP. Ketika proses perpindahan router terjadi, akan ada pencarian ARP yang dilakukan oleh komputer yang melakukan koneksi ke router (gambar 4.68). ARP ini bertujuan mencari kecocokan alamat IP dengan alamat MAC. Broadcast ARP ini akan berlangsung sampai dengan router aktif telah terpilih dengan kata lain alamat virtual IP dan MAC telah ada kembali di jaringan. Karena adanya proses pencarian ARP menyebabkan terjadi request yang berulang.

No.	Time	Source	Destination	Protocol	Info
25	2004-08-03 12:17:55.061734	10.126.10.167	10.126.10.170	ICMP	Echo (ping) request
26	2004-08-03 12:17:55.348773	10.126.10.168		ARP	Who has 10.126.10.170? Te
27	2004-08-03 12:17:55.831713	10.126.100.4		ARP	Who has 10.126.100.1? Tel
29	2004-08-03 12:17:56.008742	10.126.100.5		ARP	Who has 10.126.100.1? Tel
30	2004-08-03 12:17:56.061749	10.126.10.167		ARP	Who has 10.126.10.170? Te
31	2004-08-03 12:17:56.348751	10.126.10.168		ARP	Who has 10.126.10.170? Te
32	2004-08-03 12:17:56.831735	10.126.100.4		ARP	Who has 10.126.100.1? Tel
33	2004-08-03 12:17:57.017975	10.126.100.5		ARP	Who has 10.126.100.1? Tel
34	2004-08-03 12:17:57.061716	10.126.10.167		ARP	Who has 10.126.10.170? Te
35	2004-08-03 12:17:57.349286	10.126.10.168		ARP	Who has 10.126.10.170? Te
36	2004-08-03 12:17:57.831759	10.126.100.4		ARP	Who has 10.126.100.1? Tel
37	2004-08-03 12:17:58.008710	10.126.100.5		ARP	Who has 10.126.100.1? Tel
38	2004-08-03 12:17:58.061717	10.126.10.167		ARP	Who has 10.126.10.170? Te
39	2004-08-03 12:17:58.348704	10.126.10.168		ARP	Who has 10.126.10.170? Te
40	2004-08-03 12:17:58.831714	10.126.100.4		ARP	Who has 10.126.100.1? Tel
41	2004-08-03 12:17:59.008682	10.126.100.5		ARP	Who has 10.126.100.1? Tel
42	2004-08-03 12:17:59.061746	10.126.10.170	10.126.10.170	ICMP	Echo (ping) request

Gambar 4.68 Broadcast ARP Ketika Terjadi Perpindahan Router

BAB V

KESIMPULAN DAN SARAN

Bab ini menjelaskan tentang kesimpulan yang bisa diambil dari pembuatan Tugas Akhir ini serta saran untuk kemungkinan pengembangan aplikasi sistem selanjutnya.

5.1 Kesimpulan

1. Komputer router dengan sistem operasi Linux dapat mengimplementasikan sistem semi HSRP sebagimana halya router perangkat keras.
2. Perpindahan router dapat dilakukan secara otomatis dengan memindahkan alamat virtual IP dan MAC router.
3. Besarnya lalu lintas jaringan tidak mempengaruhi waktu perpindahan router.
4. Jumlah router dalam satu kelompok Insco untuk jumlah router lebih dari dua tidak mempengaruhi waktu perpindahan router.

5.2 Saran

1. Mengurangi proses yang dilakukan router aktif, sehingga router aktif dapat meneruskan paket klien dan memberikan respon lebih cepat setiap pesan dari router standby.
2. Melakukan enskripsi protokol yang dikirimkan sehingga keamanan data dapat terjaga. Hal ini dilakukan karena aplikasi ini menggunakan komunikasi broadcast dimana seluruh komputer pada jaringan sapat menerima semua

pesan yang dikirmkan oleh router aktif maupun standby.

3. Mengurangi jumlah data dalam protokol sehingga tidak membebani jaringan dengan cara kompresi data yang akan dikirim.
4. Pengembangan antarmuka pengguna sehingga memudahkan penggunaan terutama bagi user yang masih awam linux.

DAFTAR PUSTAKA

[1] Forouzan, Behrouz A., *TCP/IP Protocol Suite*, McGraw Hill, 2000.

[2] http://daq.trium.ca/online/software/multinet/programmers_reference

[3] http://pont.net/

[4] http://utenti.lycos.it/okarti/modules.php?name=News&file=article&sid=22

[5] http://www.faqs.org/rfcs/rfc2281.html

[6] http://www.ietf.org/html.charters/vrrp-charter.html

[7] http://www.w3.org/TR/html4/

[8] Matthew, Neil, Stones, Richard, *Professional Linux Programming*, Wrox Press, 2000.